UNIVERSIDAD DE CÁDIZ

DEPARTAMENTO DE MATEMÁTICAS

Operador de consecuencias extendido por agregadores formales para modelar bases de datos con incertidumbre

JOSÉ ANTONIO TORNÉ ZAMBRANO

TESIS DOCTORAL

Febrero 2025

Dr. Jesús Medina Moreno, Catedrático de Universidad del Departamento de Matemáticas de la Universidad de Cádiz,

HACE CONSTAR:

Que D. José Antonio Torné Zambrano, Graduado en Matemáticas, ha realizado en el Departamento de Matemáticas de la Universidad de Cádiz, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

Operador de consecuencias extendido por agregadores formales para modelar bases de datos con incertidumbre

Revisado el presente trabajo, estima que puede ser presentado al Tribunal que ha de juzgarlo. Para que conste a efectos de lo establecido en el Real Decreto 99/2011, autoriza la presentación de este trabajo en la Universidad de Cádiz.

Cádiz, a 15 de febrero de 2025

Dr. Jesús Medina Moreno

A mi familia

Agradecimientos

Recuerdo el segundo cuatrimestre del cuarto curso del grado en Matemáticas, en 2019, pocos meses antes de la entrega del Trabajo de Fin de Grado. En ese momento, ya agotado de la investigación y del arduo trabajo que implicaba, juré que nunca emprendería un doctorado. Pensaba que no estaba preparado para afrontar un desafío de tal magnitud, con un nivel de exigencia tan elevado, que exige una dedicación constante, aunque solo sea unos minutos al día. Sin embargo, fueron mis padres quienes, como siempre, me alentaron y motivaron a enfrentarme a este reto, a no rendirme ni dejarme intimidar, sino a intentarlo y juzgar luego los resultados. De igual manera, la motivación y el empuje de mi hermano fueron cruciales. Por todo ello, quiero expresar mi más sincero agradecimiento a mis padres y a mi hermano por su apoyo incondicional.

Comencé el doctorado en 2021 y, justo en julio, aprobé las oposiciones al cuerpo de enseñanza secundaria en la Comunidad de Madrid. En septiembre, me mudé a Madrid y, desde entonces, he residido en esta ciudad. Realizar un doctorado a distancia no es tarea fácil, pero mi director de tesis, Jesús, siempre estuvo pendiente para brindarme su apoyo. Las reuniones los martes a las 16:30 por *Skype* se convirtieron en una constante esencial que me permitió avanzar cada semana. Asimismo, gracias a su apoyo, tuve la oportunidad de participar en diversos congresos de matemáticas, donde pude exponer los avances de mi investigación. Agradezco profundamente a mi director por su dedicación y orientación durante todo este proceso. Por último, me gustaría agradecer a todas las personas que, de una u otra forma, han contribuido a que pudiera llevar a cabo este desafío titánico que implica un doctorado.

Índice general

English Summary xv

Resumen xxiii

1. Contexto general de la tesis doctoral **1**

1.1. Introducción, justificación del estudio y análisis crítico de antecedentes 1

1.1.1. Origen y objetivos de la lógica 7

1.1.2. Lógica difusa 12

1.1.3. El problema de la decidibilidad 15

1.1.4. Componentes de una lógica computacional 18

1.1.5. Deducción automática 20

1.1.6. Programación lógica 23

1.1.7. Programación lógica multiadjunta 30

1.1.8. Cuantificadores generalizados 36

1.1.9. Operadores formales de agregación 44

1.2. Hipótesis y objetivos 51

2. Discusión conjunta de los resultados **53**

2.1. Operador de consecuencias inmediatas con cuantificadores generalizados 53

2.2. Alcance computacional de consecuencias cuantificadas a partir de datos iniciales imperfectos 62
2.3. Técnica computacional para la obtención de consecuencias en programación lógica multiadjunta con cuantificadores generalizados 64
2.4. Operador de consecuencias inmediatas a través de operadores de promedio ponderado ordenado 72
2.5. Semántica operacional sobre programación lógica de primer orden, difusa y arbitraria 80

3. Operador de consecuencias inmediatas con cuantificadores generalizados 89

4. Alcance computacional de consecuencias cuantificadas a partir de datos iniciales imperfectos 111

5. Técnica computacional para la obtención de consecuencias en programación lógica multiadjunta con cuantificadores generalizados 125

6. Operador de consecuencias inmediatas a través de operadores de promedio ponderado ordenado 147

7. Semántica operacional sobre programación lógica de primer orden, difusa y arbitraria 167

8. Conclusiones y trabajo futuro 209
8.1. Conclusiones 209
8.2. Trabajo futuro 215

Anexos. Otras publicaciones relacionadas 217

A. ***Fuzzy Logic Programming with Generalized Quantifiers*** **219**

B. ***Preference in Multi-Adjoint Logic Programming based on Ordered Adjoint Pairs*** **227**

C. ***Hypergraph logic program representation versus stratified programs*** **235**

D. ***Revisiting the immediate consequences operator on first-order logic*** **243**

Bibliografía **255**

English Summary

Logic Programming was born from the work carried out on automated theorem proving and artificial intelligence at the beginning of the second half of the 20th century, and among them it stands out the work of Robinson in which he introduced a very useful inference rule for the implementation of the automation process in computers. It was in 1972 when the first Logic Programming interpreter was formally implemented and studied, called `ProLog`, derived from the French *PROgrammation en LOGique*, and based on the work carried out by the teams of both Robert Kowalski and Alain Colmerauer. The use of logic as a programming language was a revolutionary idea and, since then, different logic programming paradigms have been presented.

Among the extensions of traditional logic programming, Multi-adjoint Logic Programming stands out. This extension of both Fuzzy Logic Programming and Monotonic and Residuated Logic Programming allows a wide variety of different types of logical connectives and adjoint pairs to be considered in the construction of the logic rules of the program. This is not the only advantage of the multi-adjoint environment; additionally, logical connectors are required to satisfy fewer properties, and can therefore be significantly more flexible and better suited to the input data. This enables a more realistic approach to modeling expert systems grounded in logical rules.

The semantics of a multi-adjoint logic program is characterized by the

immediate consequences operator which, in a fuzzy environment, is defined as the supremum of the values provided by each of the logical rules. However, the rigid nature of the supremum, together with a, more than likely, poor quality of the data modeling the logical rules of the program, may result in unreliable values to be obtained in the aggregation. This high sensitivity of the supremum operator to small errors or variations in the data, in addition to leading to an evident loss of information, can have drastic consequences in a decision-making process based on the conclusions drawn from expert systems modeled by these logical rules.

Consequently, to provide a solution to this problem, the main objective of this thesis is to introduce an alternative definition of the immediate consequences operator based on formal aggregation operators, and, subsequently, to study its properties and propose new logic programming semantics built upon this operator. In particular, the immediate consequences operator will be extended through generalized quantifiers and ordered weighted average operators, which were introduced by Yager.

This thesis has been presented as a compendium of four papers that have been published in different journals, all of them in the first quartile of their category:

- *Immediate consequences operator on generalized quantifiers* (Fuzzy Sets and Systems) [129].
- *Computational reaching of quantified consequences from imperfect initial data* (Mathematical Methods in the Applied Sciences) [122].
- *Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers* (Applied and Computational Mathematics) [131].
- *Immediate consequences operator through ordered weighted average operators* (Iranian Journal of Fuzzy Systems) [130].

Below, we provide a detailed overview of the content of each of the chapters of the thesis.

First of all, in Chapter 1 we will address issues related to the general context of the thesis. Through an introduction and a detailed analysis of all the background that has allowed the development of Multi-adjoint Logic Programming (the study of classical logic, fuzzy logic, automated deduction, logic programming, among other aspects, will be carried out), the environment in which the thesis is developed, we will justify the need to extend the immediate consequences operator. In addition, we will present the key developments made regarding the theory of generalized quantifiers and ordered weighted average operators, which will help us to understand the advantages of these operators to relax the strict nature of the supremum in the definition of the immediate consequences operator.

Next, in Chapter 2, a joint discussion of the results achieved in each of the published papers is carried out. We have proceeded by analyzing each article following a logical order that enhances understanding and enables a coherent and structured analysis of the topics discussed.

In the paper *Immediate consequences operator on generalized quantifiers* [129], we take advantage of the flexibility provided by generalized quantifiers to weaken the strict nature of the supremum operator in the definition of the immediate consequences operator, with the aim of reducing the potential noise present in the data. In addition, interesting properties such as monotonicity are studied, and a new, more versatile fixed point semantics is proposed.

The work done in the previous paper is extended in the following two papers. In order to computationally derive the precise consequences of a dataset modeled by logical rules, one of the key properties that the new definition of the immediate consequences operator must satisfy is continuity. This property ensures that the information and consequences derived from the database, can be obtained computationally through the

quantified immediate consequences operator in a countable number of iterations. Specifically, in *Computational reaching of quantified consequences from imperfect initial data* [122], necessary and sufficient conditions are studied to ensure this continuity in the new flexible version in which we are working.

On the other hand, in the paper *Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers* [131] an operational semantics is studied in order to implement an automatic reasoning procedure that enables the retrieval of information and consequences from knowledge-based systems modeled by logical rules with the possible presence of inaccuracy in the data.

In the paper *Immediate consequences operator through ordered weighted average operators* [130], it is introduced a new generalization of the immediate consequences operator based on ordered weighted average operators, with the objective of decreased the possible loss of information derived from environments with uncertainty. Furthermore, a new fixed point semantics adapted to this more versatile environment is proposed.

As a final point, in the paper *Operational semantics on arbitrary first-order fuzzy logic programming* [128], we propose a new operational semantics that is adaptable to any monotonic extension of the immediate consequences operator, within the context of multi-adjoint first-order logic programming.

As we mentioned above, the first four papers have been published in high-impact scientific journals, in four journals listed in the *Journal Citation Reports*, JCR in short version. All of them are in the first quartile (Q1) of their category in the JCR. The last paper has been submitted to a high-impact (Q1) journal for publication. These papers can be found in Chapters 3, 4, 5, 6 and 7 respectively. In Annexes A, B, C and D are the published book chapters.

Finally, in Chapter 8, we present the most relevant conclusions drawn,

along with the research directions that remain as future work.

Publications

Many of the results presented in this thesis have already been published in papers in journals or at conferences:

1. J. Medina and J. A. Torné-Zambrano. Immediate consequences operator on generalized quantifiers. *Fuzzy Sets and Systems*, 456, 2022. I.S.S.N: 0165-0114. It is found in Chapter 3.
    - Impact Factor: 3.9.
    - Position at the JCR: 12 de 267 (Q1). Category: MATHEMATICS, APPLIED. JCR Year: 2022.
2. J. Medina and J. A. Torné-Zambrano. Computational reaching of quantified consequences from imperfect initial data. *Mathematical Methods in the Applied Sciences*, 2024. I.S.S.N: 0170-4214. It is found in Chapter 4.
    - Impact Factor: 2.1.
    - Position at the JCR: 53 de 332 (Q1). Category: MATHEMATICS, APPLIED. JCR Year: 2023.
3. J. Medina and J. A. Torné-Zambrano. Computational Technique for Obtaining Consequences in Multi-adjoint Logic Programming with Generalized Quantifiers. *Applied and Computational Mathematics*, 2024. I.S.S.N: 1683-3511. It is found in Chapter 5.
    - Impact Factor: 4.6.
    - Position at the JCR: 2 de 332 (Q1). Category: MATHEMATICS, APPLIED. JCR Year: 2024.

4. J. Medina and J. A. Torné-Zambrano. Immediate consequences operator through ordered weighted average operators. *Iranian Journal of Fuzzy Systems*, 20(5), 2023. I.S.S.N: 1735-0654. It is found in Chapter 6.

    - Impact Factor: 1.9.
    - Position at the JCR: 34 de 490 (Q1). Category: MATHEMATICS. JCR Year: 2023.

The following paper has been submitted to a high-impact journal:

5. J. Medina and J. A. Torné-Zambrano. Operational semantics on arbitrary first-order fuzzy logic programming. It is found in Chapter 7.

Other publications are:

6. J. Medina and J. A. Torné-Zambrano. Fuzzy Logic Programming with Generalized Quantifiers. In Cornejo, M. E., Harmati, I. Á., Kóczy, L. T., Medina, J., editors, *Computational Intelligence and Mathematics for Tackling Complex Problems 4. Studies in Computational Intelligence*, volume 1040, pages 17–23. Springer International Publishing, Cham, 2023. It is found in Annex A.

7. J. Medina and J. A. Torné-Zambrano. Preference in Multi-adjoint Logic Programming Based on Ordered Adjoint Pairs. In Cornejo, M., Kóczy, L.T., Medina, J., Ramírez-Poussa, E., editors, *Computational Intelligence and Mathematics for Tackling Complex Problems 5. Studies in Computational Intelligence*, volume 1127, pages 93-99. Springer Nature Switzerland, Cham, 2024. It is found in Annex B.

8. D. Lobo, J. Medina, J. R. Portillo and J. A. Torné-Zambrano. Hypergraph logic program representation versus stratified programs. In Kóczy, L.T., Kowalski, P. A., Medina, J., Ramírez-Poussa, E., editors,

Computational Intelligence and Mathematics for Tackling Complex Problems 6. Studies in Computational Intelligence. Springer Nature Switzerland, Cham, 2025. In press. It is found in Annex C.

9. J. Medina and J. A. Torné-Zambrano. Revisiting the immediate consequences operator on first-order logic. In Lesot, M.J., Vieira, S., Reformat, M. Z., Carvalho, J. P., Batista, F., Bouchon-Meunier, B., Yager, R. R., editores, *Information Processing and Management of Uncertainty in Knowledge-Based Systems. IPMU 2024. Lecture Notes in Networks and Systems*, volume 1176. Springer Nature Switzerland, Cham, 2025. In press. It is found in Annex D.

Contributions to mathematics conferences:

10. J. Medina and J. A. Torné-Zambrano. *Generalized quantifiers to multi-adjoint fixed point semantics*. In Computational and Mathematical Methods in Science and Engineering (CMMSE2021), 2021.

11. J. Medina y J. A. Torné-Zambrano. *Towards fuzzy logic programming with generalized quantifiers*. In Boffa, S., Ciucci, D., Couso, I., Medina, J., Ślęzak, D., editors, Book of Abstracts from the 19th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU2022), pages 31-32, 2022.

12. J. Medina and J. A. Torné-Zambrano. *A non-deterministic procedure for query answering with generalized quantifiers*. In Computational and Mathematical Methods in Science and Engineering (CMMSE2023), 2023.

13. J. Medina y J. A. Torné-Zambrano. *On generalized quantifiers in multi-adjoint logic programming*. In Massanet, S., Montes, S., Ruiz-Aguilera, D., González-Hidalgo, M., editors, Book of Abstracts from the 13th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT2023), page 45, 2023.

14. J. Medina and J. A. Torné-Zambrano. *A procedural semantics to generalized immediate consequences operator*. In Computational and Mathematical Methods in Science and Engineering (CMMSE2024), 2024.

15. D. Lobo, J. Medina, J. R. Portillo y J. A. Torné-Zambrano. *Stratified logic programs through hypergraphs*. In 15th European Symposium on Computational Intelligence and Mathematics (ESCIM2024), 2024.

Resumen

La Programación Lógica nació a partir de los trabajos realizados en demostración automática de teoremas e inteligencia artificial a principios de la segunda mitad del siglo XX, entre los que destacó el trabajo de Robinson, donde introdujo una regla de inferencia que resultó fundamental para el desarrollo de la Programación Lógica, ya que permitió la realización del proceso de automatización en los ordenadores. Fue en 1972 cuando se implementó y estudió formalmente el primer intérprete de la Programación Lógica, llamado `ProLog`, derivado del francés *PROgrammation en LOGique*, a partir de los estudios desarrollados por los equipos tanto de Robert Kowalski como de Alain Colmerauer. El uso de la lógica como un lenguaje de programación supuso una idea revolucionaria y, desde entonces, se han presentado diferentes paradigmas de programación lógica.

Entre las extensiones de la programación lógica tradicional destaca la Programación Lógica Multiadjunta. Esta extensión, tanto de la Programación Lógica Difusa como de la Programación Lógica Monótona y Residuada, permite considerar una gran variedad de diferentes tipos de conectores lógicos y pares adjuntos en la construcción de las reglas lógicas del programa. Esta no es la única ventaja que presenta el entorno multiadjunto, sino que a los conectores lógicos se les exigen menos propiedades, pudiendo ser, por lo tanto, mucho más flexibles y adaptarse mejor a los datos de partida. Todo ello posibilita un enfoque más realista para modelizar sistemas expertos a partir de reglas lógicas.

La semántica de un programa lógico multiadjunto se caracteriza a partir del operador de consecuencias inmediatas que, en un ambiente difuso, se define como el supremo de los valores aportados por cada una de las reglas. No obstante, el carácter rígido del supremo unido a una, más que probable, mala calidad de los datos que modelan las reglas lógicas del programa, puede provocar que se obtengan valores que no sean fiables en la agregación. Esta enorme sensibilidad del supremo a pequeños errores o variaciones en los datos, además de conllevar una evidente pérdida de información, puede ocasionar consecuencias drásticas en un proceso de toma de decisiones a partir de las conclusiones alcanzadas en sistemas expertos modelados por estas reglas lógicas.

En consecuencia, para dar respuesta a esta problemática, el principal objetivo de la presente tesis es introducir una definición alternativa del operador de consecuencias inmediatas basada en operadores formales de agregación, y, posteriormente, estudiar sus propiedades y proponer nuevas semánticas de programación lógica articuladas a partir de este operador. En particular, el operador de consecuencias inmediatas se extenderá a partir de los cuantificadores generalizados y los operadores de promedio ponderado ordenado que introdujo Yager.

Esta tesis se ha presentado como compendio de cuatro artículos que se han publicado en distintas revistas, todas ellas en el primer cuartil de su categoría:

- *Immediate consequences operator on generalized quantifiers* (Fuzzy Sets and Systems) [129].
- *Computational reaching of quantified consequences from imperfect initial data* (Mathematical Methods in the Applied Sciences) [122].
- *Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers* (Applied and Computational Mathematics) [131].

- *Immediate consequences operator through ordered weighted average operators* (Iranian Journal of Fuzzy Systems) [130].

A continuación, se expone el contenido de cada uno de los capítulos de la tesis.

En primer lugar, en el Capítulo 1 se abordarán cuestiones relativas al contexto general de la tesis. A través de una introducción y un análisis pormenorizado de todos los antecedentes que han permitido el desarrollo de la Programación Lógica Multiadjunta (se llevará a cabo el estudio de la lógica clásica, la lógica difusa, la deducción automática, la programación lógica, entre otros aspectos), entorno en el que se desarrolla la tesis, justificaremos la necesidad de extender el operador de consecuencias inmediatas. Asimismo, expondremos las principales investigaciones que se han realizado con respecto a la teoría de los cuantificadores generalizados y los operadores de promedio ponderado ordenado, lo cual permitirá entender las ventajas de estos operadores para flexibilizar el carácter estricto del supremo en la definición del operador de consecuencias inmediatas.

Posteriormente, en el Capítulo 2, se realiza un análisis detallado de los resultados alcanzados en cada uno de los artículos publicados. Hemos procedido analizando cada artículo siguiendo un orden lógico que facilita su comprensión y permite un análisis coherente y gradual de los temas tratados.

En *Immediate consequences operator on generalized quantifiers* [129], nos valemos de la flexibilidad que aportan los cuantificadores generalizados para debilitar el carácter estricto del operador supremo en la definición del operador de consecuencias inmediatas, con el objetivo de reducir el posible ruido presente en los datos. Posteriormente, se estudian propiedades interesantes como la monotonía, y se propone una nueva semántica de punto fijo más versátil.

El trabajo realizado en este artículo se continúa con los dos siguien-

tes. Para poder obtener computacionalmente las consecuencias precisas de un conjunto de datos modelados por reglas lógicas, uno de los requisitos fundamentales que debe cumplir la nueva definición del operador de consecuencias inmediatas es la continuidad, puesto que garantiza que la información y consecuencias derivadas de la base de datos modelada por reglas lógicas puede obtenerse computacionalmente a partir del operador de consecuencias inmediatas cuantificado en un cantidad numerable de iteraciones. En concreto, en *Computational reaching of quantified consequences from imperfect initial data* [122], se estudian condiciones necesarias y suficientes para asegurar esta continuidad en la nueva versión flexible en la que nos encontramos.

Por otro lado, en el artículo *Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers* [131] se estudia una semántica operacional para poder implementar un procedimiento de razonamiento automático que permita obtener información y consecuencias de sistemas de conocimientos modelados por reglas lógicas con la posible presencia de imprecisión en los datos.

En el artículo *Immediate consequences operator through ordered weighted average operators* [130], introducimos una nueva generalización del operador de consecuencias inmediatas basada en operadores de promedio ponderado ordenado, con el objetivo de reducir la posible pérdida de información derivada de ambientes con incertidumbre. Además, se propone una nueva semántica de punto fijo adaptada a este entorno más versátil.

Finalmente, en el artículo *Operational semantics on arbitrary first-order fuzzy logic programming* [128], proponemos una nueva semántica operacional adaptable a cualquier extensión del operador de consecuencias inmediatas que sea monótona, para un contexto de programación lógica de primer orden multiadjunta.

Los primeros cuatro artículos se han publicado en revistas científicas de alto impacto, en cuatro revistas recogidas en el *Journal Citation Reports*,

JCR de forma abreviada. Todos ellos se sitúan en el primer cuartil (Q1) de su categoría del JCR. El última artículo se ha enviado a una revista de impacto (Q1) para su publicación. Estos artículos se pueden encontrar en los Capítulos 3, 4, 5, 6 y 7 respectivamente. En los Anexos A, B, C y D se hallan los capítulos de libros publicados.

Finalmente, en el Capítulo 8, presentamos las conclusiones más relevantes alcanzadas, junto con las líneas de investigación que quedan pendientes como trabajo futuro.

Publicaciones

Muchos de los resultados presentados en esta tesis ya han sido publicados en artículos en revistas o en congresos:

1. J. Medina y J. A. Torné-Zambrano. Immediate consequences operator on generalized quantifiers. *Fuzzy Sets and Systems*, 456, 2022. I.S.S.N: 0165-0114. Se encuentra en el Capítulo 3.

    - Factor de Impacto: 3,9.
    - Posición en el JCR: 12 de 267 (Q1). En el campo: MATHEMATICS, APPLIED. Año JCR: 2022.

2. J. Medina y J. A. Torné-Zambrano. Computational reaching of quantified consequences from imperfect initial data. *Mathematical Methods in the Applied Sciences*, 2024. I.S.S.N: 0170-4214. Se encuentra en el Capítulo 4.

    - Factor de Impacto: 2,1.
    - Posición en el JCR: 53 de 332 (Q1). En el campo: MATHEMATICS, APPLIED. Año JCR: 2023.

3. J. Medina y J. A. Torné-Zambrano. Computational Technique for Obtaining Consequences in Multi-adjoint Logic Programming with Generalized Quantifiers. *Applied and Computational Mathematics*, 2024. I.S.S.N: 1683-3511. Se encuentra en el Capítulo 5.

    - Factor de Impacto: 4,6.
    - Posición en el JCR: 2 de 332 (Q1). En el campo: MATHEMATICS, APPLIED. Año JCR: 2024.

4. J. Medina y J. A. Torné-Zambrano. Immediate consequences operator through ordered weighted average operators. *Iranian Journal of Fuzzy Systems*, 20(5), 2023. I.S.S.N: 1735-0654. Se encuentra en el Capítulo 6.

    - Factor de Impacto: 1,9.
    - Posición en el JCR: 34 de 490 (Q1). En el campo: MATHEMATICS. Año JCR: 2023.

El siguiente artículo se ha enviado para su revisión a una revista de impacto:

5. J. Medina y J. A. Torné-Zambrano. Operational semantics on arbitrary first-order fuzzy logic programming. Se encuentra en el Capítulo 7.

Otras publicaciones son:

6. J. Medina y J. A. Torné-Zambrano. Fuzzy Logic Programming with Generalized Quantifiers. En Cornejo, M. E., Harmati, I. Á., Kóczy, L. T., Medina, J., editores, *Computational Intelligence and Mathematics for Tackling Complex Problems 4. Studies in Computational Intelligence*, volumen 1040, páginas 17–23. Springer International Publishing, Cham, 2023. Se encuentra en el Anexo A.

7. J. Medina y J. A. Torné-Zambrano. Preference in Multi-adjoint Logic Programming Based on Ordered Adjoint Pairs. En Cornejo, M., Kóczy, L.T., Medina, J., Ramírez-Poussa, E., editores, *Computational Intelligence and Mathematics for Tackling Complex Problems 5. Studies in Computational Intelligence*, volumen 1127, páginas 93-99. Springer Nature Switzerland, Cham, 2024. Se encuentra en el Anexo B.

8. D. Lobo, J. Medina, J. R. Portillo y J. A. Torné-Zambrano. Hypergraph logic program representation versus stratified programs. En Kóczy, L.T., Kowalski, P. A., Medina, J., Ramírez-Poussa, E., editores, *Computational Intelligence and Mathematics for Tackling Complex Problems 6. Studies in Computational Intelligence*. Springer Nature Switzerland, Cham, 2025. Pendiente de publicación. Se encuentra en el Anexo C.

9. J. Medina y J. A. Torné-Zambrano. Revisiting the immediate consequences operator on first-order logic. En Lesot, M.J., Vieira, S., Reformat, M. Z., Carvalho, J. P., Batista, F., Bouchon-Meunier, B., Yager, R. R., editores, *Information Processing and Management of Uncertainty in Knowledge-Based Systems. IPMU 2024. Lecture Notes in Networks and Systems*, volumen 1176. Springer Nature Switzerland, Cham, 2025. Pendiente de publicación. Se encuentra en el Anexo D.

Aportaciones a congresos de matemáticas:

10. J. Medina y J. A. Torné-Zambrano. *Generalized quantifiers to multi-adjoint fixed point semantics*. En Computational and Mathematical Methods in Science and Engineering (CMMSE2021), 2021.

11. J. Medina y J. A. Torné-Zambrano. *Towards fuzzy logic programming with generalized quantifiers*. En Boffa, S., Ciucci, D., Couso, I., Medina, J., Ślęzak, D., editores, Libro de Resúmenes del Congreso 19th International Conference on Information Processing and Management of

Uncertainty in Knowledge-Based Systems (IPMU2022), páginas 31-32, 2022.

12. J. Medina y J. A. Torné-Zambrano. *A non-deterministic procedure for query answering with generalized quantifiers*. En Computational and Mathematical Methods in Science and Engineering (CMMSE2023), 2023.

13. J. Medina y J. A. Torné-Zambrano. *On generalized quantifiers in multi-adjoint logic programming*. En Massanet, S., Montes, S., Ruiz-Aguilera, D., González-Hidalgo, M., editores, Libro de Resúmenes del Congreso 13th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT2023), página 45, 2023.

14. J. Medina y J. A. Torné-Zambrano. *A procedural semantics to generalized immediate consequences operator*. En Computational and Mathematical Methods in Science and Engineering (CMMSE2024), 2024.

15. D. Lobo, J. Medina, J. R. Portillo y J. A. Torné-Zambrano. *Stratified logic programs through hypergraphs*. En 15th European Symposium on Computational Intelligence and Mathematics (ESCIM2024), 2024.

Capítulo 1

Contexto general de la tesis doctoral

1.1. Introducción, justificación del estudio y análisis crítico de antecedentes

En el último siglo se han producido numerosos avances sociales, económicos, ambientales, políticos, etc., destacando, principalmente, los progresos realizados en el ámbito de la tecnología y la ciencia, que han transformado inconscientemente la realidad en la que vivimos de una manera drástica, veloz y, definitivamente, significativa. Este progreso tecnológico ha impactado en todos los ámbitos de la vida de las personas, desde el aspecto económico y laboral, hasta el social. Actualmente, la ciencia y la tecnología han cambiado la manera en la que se realizan las investigaciones científicas, la forma como se comunican las familias, el modo en que nos desplazamos; incluso han repercutido en aspectos de seguridad, toma de decisiones y acceso a la información. Antiguamente, y estamos hablando de hace 30 años, hasta que no leías un periódico o veías las noticias en el televisor, no te enterabas de lo que estaba ocurriendo, por ejemplo, en

Oriente Medio. Hoy en día basta con realizar una búsqueda en el navegador de tu teléfono móvil o *smartphone*, sin importar prácticamente el lugar donde te encuentres. El cambio, siempre que su uso sea el adecuado, es, por supuesto, enriquecedor.

Se lograron progresos en las redes y las telecomunicaciones, como el nacimiento de ARPANET (*Advanced Research Projects Agency NETwork*) en 1969 como precursor directo del Internet; en 1957 se produjo el lanzamiento del *Sputnik*, dando el pistoletazo de salida a la carrera espacial y al desarrollo de satélites artificiales; en 1946 se lanzó el primer ordenador electrónico digital de propósito general, ENIAC (*Electronic Numerical Integrator And Computer*); se introdujeron, a partir de 1970, sistemas de gestión de bases de datos relacionales, RDBMS (*Relational DataBase Management System*), como *Oracle* o *IBM System R*; a finales de la década de 1980 se desarrolla la *World Wide Web*, el primer sistema de distribución global de información a través de Internet, utilizando el Protocolo de Transferencia de Hipertextos, HTTP; se desarrollaron los primeros navegadores webs, *WorldWideWeb* (1990), *ViolaWWW* (1992), *Mosaic* (1993), *Netscape* (1994), *Internet Explorer 1.0* (1995); en 2006 se lanzaron al público productos de *Amazon Web Services*, *Amazon Simple Storage Service* y *Elastic Compute Cloud*, siendo de los primeros sistemas de almacenamiento en la nube, para guardar grandes cantidades de datos; a partir de principios del siglo XXI, con particular énfasis durante la década de 2010, se monitorizan los sistemas de distintos aparatos como dispositivos de seguridad, electrodomésticos, pulseras y relojes, luces, termostatos, etc., a través del internet de las cosas, *Internet of Things (IoT)*; proliferan las redes sociales como *Six Degrees* (1997), *Friendster* (2002), *MySpace* (2003), *LinkedIn* (2003), *Facebook* (2004). Estos avances revolucionarios impulsaron un crecimiento exponencial en la generación de datos, *Big Data* en su notación inglesa, debido, principalmente, a una digitalización masiva y a la proliferación de grandes cantidades de información, obtenidas a través de los datos de internet, de las redes sociales,

de las diferentes aplicaciones y dispositivos móviles, de los sensores, etc. Además, se diseñaron mejores ordenadores, con una capacidad computacional mucho mayor, lo que, junto con grandes avances en el campo del aprendizaje automático, los convertían en herramientas idóneas para procesar, tratar, almacenar y transmitir grandes cantidades de datos, haciendo posible transformar los vastos volúmenes de información en conocimiento. Las empresas de casi cualquier industria supieron aprovecharse de estos avances tecnológicos para explotar los datos con el objetivo de obtener beneficios competitivos [157].

Todos estos avances se fusionaron e interrelacionaron, dando lugar a una nueva rama científica interdisciplinar, la Ciencia de Datos, llamada en inglés *Data Science*, cuyo objetivo es analizar y transformar enormes volúmenes de datos en información útil para diferentes finalidades como la toma de decisiones, la predicción de tendencias o la optimización de procesos y sistemas. Como establecen los científicos de datos Foster Provost y Tom Fawcett, "*Data Science involves principles, processes, and techniques for understanding phenomena via the (automated) analysis of data*", es decir, "la Ciencia de Datos integra principios, procesos y técnicas para comprender fenómenos a través del análisis (automatizado) de datos" [157].

El aforismo "El conocimiento es poder", generalmente atribuida al filósofo inglés Francis Bacon, nunca había sido más cierto que hoy en día, considerando que a través de los datos se puede extraer información para ofrecer diversos servicios personalizados a los usuarios, ayudar a la toma de decisiones empresariales basadas en las tendencias del mercado, optimizar operaciones, generar publicidad adaptada a cada necesidad particular, identificar patrones en enfermedades y proponer mejores tratamientos, adoptar políticas medioambientales basadas en hechos y ofrecer soluciones, prevenir el crimen y la delincuencia, desarrollar infraestructuras, etc. La relevancia de las bases de datos también se manifiesta en las investigaciones científicas, la minería de datos, la inteligencia artificial, el modelaje

de sistemas complejos, el análisis de riesgos, la predicción de precios de activos, la detección de fraude en servicios financieros, el marketing, la optimización de rutas y recursos, la recomendación de productos en el comercio electrónico, los procesos de automatización industriales, etc. Todos estos ejemplos evidencian la importancia de los datos y, por ende, de las bases de datos [33, 176].

No obstante, no todos los datos son útiles para obtener información significativa. Como menciona Nate Silver en [179], "si la cantidad de información aumenta a razón de 2,5 trillones de bytes por día, es casi seguro que la cantidad de información útil no lo hará. La mayor parte es solo ruido, y el ruido aumenta más rápido que la señal", donde el autor utiliza una metáfora para referirse a la información útil del conjunto de datos como *señal*, mientras que el *ruido* es la información irrelevante, falsa o con errores que dificulta la identificación de las señales. La presencia de ruido en los datos es uno de los problemas más destacados dentro del ámbito del *Big Data* y, por extensión, de la Ciencia de Datos, siendo el objetivo mejorar la calidad de los datos. La noción de *calidad de los datos* "tiene graves consecuencias, de gran trascendencia, en la eficiencia y efectividad de organizaciones y empresas" [14]. Los primeros pasos en el área de la calidad de los datos fueron dados en el ámbito de la estadística, a finales de los años 60 del siglo XX, proponiendo teorías matemáticas para eliminar datos duplicados en los conjuntos estadísticos. Sin embargo, el problema de la calidad de los datos no adquirió mayor relevancia hasta la década de 1990, cuando los investigadores de la Ciencia de la Computación empezaron a abordar de manera formal los desafíos de definir, medir y mejorar la calidad de los datos almacenados en bases de datos [14].

Existen infinidad de ejemplos y casos reales relacionados con la presencia de ruido en los datos. Por ejemplo, la pérdida o retraso en la entrega de un correo puede no ser debido a un mal funcionamiento del servicio postal sino a la presencia de errores en las bases de datos de las direcciones.

Siguiendo lo establecido en [14], podríamos tener una base de datos sobre una plataforma de contenidos audiovisuales como la que se expone de manera sintetizada en la Tabla 1.1, aunque esta base de datos también podría haberse referido a la gestión de cuentas bancarias, el almacenamiento y procesamiento de transacciones financieras, el registro civil, las existencias de cualquier tienda, el registro de clientes de un comercio electrónico, el almacenamiento de historiales médicos electrónicos, la coordinación de horarios y rutas en aerolíneas, el almacenamiento de perfiles de usuarios de redes sociales, etc. En la Tabla 1.1 podemos detectar la presencia de rui-

Título	Director	Año	Presupuesto	Recaudación	Duración	IMDb	Filmaffinity	Rotten Tomatoes
The Thing	Peckinpah	1982	15M	19.6M	109	8.2	7.3	85 %
The Warriors	Hill	1979	4M		93	7.5	7	88 %
Blde Ranner	Scott	1982	28M	33.8M	117	8.1	8.1	89 %
Black Hawk Down	Scott	2026	92M	173M	144	7.7	6.9	77 %
The Wild Bunch	Carpenter	1969	6M	15.7M	143	7.9	8	91 %
Serpico	Lumet	1973	3M	29.8M		7.7	7.3	93 %

Tabla 1.1: Base de datos de una plataforma de contenidos audiovisuales. El presupuesto y la recaudación se miden en millones de dólares, mientras que la duración está en minutos.

do en los datos. En primer lugar, hay errores relacionados con la precisión de los datos. El título de la tercera película es "*Blade Runner*", pero está escrito mal. Además, los directores de las películas "*The Thing*" y "*The Wild Bunch*" están intercambiados. Otro problema estaría relacionado con los datos incompletos, como la recaudación de "*The Warriors*" o la duración de "*Serpico*". También hay un problema de inconsistencia con el año de estreno de "*Black Hawk Down*". Finalmente, los valores de las puntuaciones de las películas que reseñan las distintas plataformas de *rating* necesitan irse actualizando, a medida que los usuarios y los críticos las valoran.

Desde la década de 1990, se ha hecho evidente la problemática deriva-

da de la presencia de ruido en los datos, así como la elevada cantidad de errores en las bases de datos. En [199], los investigadores de Ciencia de Datos Richard Y. Wang y Diane M. Strong mencionan que más del 60 % de las empresas consultadas durante un estudio sobre la industria, siendo 500 corporaciones medianas con ventas anuales que ascienden a más de 20 millones de dólares, tienen problemas con la calidad de los datos. Además, en [77] se menciona que la presencia de ruido en los datos, es decir, de una calidad de los datos bastante deficiente, puede causar que en áreas como el aprendizaje automático (*machine learning*, en inglés), el tiempo necesario para realizar el aprendizaje y la complejidad de los modelos se incrementen. En [167] se hace referencia a que los errores en los datos pueden causar "pérdidas de ingresos, ineficiencia del proceso e incumplimiento de las regulaciones industriales y gubernamentales". Según [14], se estima que los datos erróneos cuestan a las empresas estadounidenses mil millones de dólares al año. Además, [167] establece que las empresas suelen tener una tasa de error de datos de aproximadamente el 1 % o el 5 %, aunque algunas llegan a superar el 30 %. Por lo tanto, es evidente que la presencia de ruido en los datos puede llevar a decisiones erróneas que, en ciertas circunstancias, pueden tener consecuencias graves.

Es posible extraer conclusiones y derivar consecuencias a partir de bases de datos mediante su modelado utilizando reglas lógicas y, por tanto, la programación lógica [118]. Entre las diversas generalizaciones de esta rama de la lógica, una de las más destacadas es la programación lógica multiadjunta [124], que es una extensión de la programación lógica difusa [196] y de la programación lógica monótona y residuada, que a su vez generalizan otros enfoques [44]. La semántica de un programa lógico, es decir, las principales consecuencias que se deducen del mismo, se caracteriza usualmente a partir de los puntos posfijos del operador de consecuencias inmediatas, el cual, en el marco multiadjunto, se define como el supremo del conjunto de valores aportados por las reglas [124]. No obs-

tante, la presencia de ruido en los datos, es decir, de información errónea, puede llevar a resultados que no sean fiables al aplicar el operador de consecuencias inmediatas. En este contexto, el principal objetivo de la tesis es reducir el impacto del operador supremo, flexibilizando la definición del operador de consecuencias inmediatas mediante el uso de cuantificadores generalizados [59] y operadores promedio ponderados ordenados [209], los cuales ofrecen herramientas más potentes para manejar la incertidumbre, la imprecisión y el ruido presente en las bases de datos.

1.1.1. Origen y objetivos de la lógica

Desde tiempos inmemoriales, el ser humano siempre ha intentado encontrar respuestas al inmenso y misterioso mundo que nos rodea. Esta respuesta debe ser dada de una manera racional, de forma que nuestro cerebro sea capaz de *comprender* el razonamiento realizado. Pero, para ello, necesitamos disponer de *algo* que sea capaz de organizar y sistematizar nuestra capacidad de razonamiento.

El intento de estructurar el razonamiento humano sentó las bases de la lógica como la disciplina encargada, no solo de determinar la veracidad de las afirmaciones, sino también de discernir aquello que es auténticamente verdadero. Como señalan Irving Copi y Carl Cohen en [40], "la lógica se ocupa del estudio de los principios y métodos utilizados para distinguir el razonamiento correcto del incorrecto".

Aunque algunas civilizaciones, como la china o la india, y filósofos de la Antigua Grecia, como Sócrates, Zenón de Elea, Parménides o Platón, ya trabajaron aspectos relacionados con la lógica, fue el tratado *Órganon*, de Aristóteles, en el siglo IV a. C., la obra que concedió un rango más formal al estudio de la lógica, erigiendo los fundamentos de esta disciplina. Entre otras cosas, esta obra proporcionó un principio de inferencia correcto como el silogismo, que permite derivar conclusiones válidas a partir de

premisas dadas. A través de estos trabajos se estableció que era posible analizar la verdad de un argumento para llegar a verdades inmutables. Si las diferentes ideas relacionadas con un argumento son claras y están lógicamente conectadas entonces, a través del razonamiento, podemos llegar a una conclusión sobre la veracidad de las premisas iniciales. De esta forma, la lógica se convierte en una herramienta muy poderosa para poder establecer verdades en un mundo de incertidumbre.

Pronto, la lógica se convirtió en el corazón de la ciencia y, en particular, de las matemáticas. Tras las contribuciones de los lógicos de la Edad Media, como Boecio, Pedro Abelardo o Guillermo de Ockham, no fue hasta principios del siglo XX cuando se desarrolló una lógica más formal, la *lógica matemática*. Diferentes avances hicieron posible esta estructuración de la lógica [110], a saber, los trabajos de Augustus De Morgan en el álgebra abstracta entre los años 1838 y 1844 [47], junto con sus trabajos en los fundamentos de la lógica formal [49, 48]; el desarrollo en 1847 del álgebra booleana por George Boole [22, 21], junto con la concepción de un sistema general para la expresión y manipulación de proposiciones, es decir, afirmaciones sobre algo. De esta forma, los trabajos de Boole supusieron un avance considerable hacia el objetivo que se planteó Gottfried Leibniz de desarrollar un lenguaje simbólico matemáticamente exacto, en el que los conceptos pudieran expresarse de manera clara. El mismo Leibniz pensó una especie de aparato lógico con el que poder realizar operaciones lógicas de forma eficaz y precisa [40]. Además, también hay que mencionar la teoría de conjuntos de Georg Cantor a partir del año 1874 [26]; la axiomatización de la aritmética por Giuseppe Peano en 1889 [149], que ayudó a crear un sistema formal para el estudio de las matemáticas, al cual se podía reducir la lógica, junto con sus trabajos en la formalización de la notación lógica [150]; y las contribuciones de los lógicos y matemáticos Charles Sanders Peirce, David Hilbert, John Venn, Ernst Schröder, Jan Łukasiewicz y Alfred Tarski, entre otros [19]. Sin embargo, fue gracias a los esfuerzos

y trabajos de los lógicos Gottlob Frege en *Begriffsschrift* [64], y Bertrand Russell y Alfred North Whitehead en *Principia Mathematica* [200], que se consiguió formalizar una lógica simbólica.

Esta lógica moderna que estaba de acuerdo con los postulados de Frege, Russell y North Whitehead sobre la validez de los argumentos, comúnmente llamada *clásica* o *estándar*, cumple las siguientes propiedades, que la definen [22, 40, 188, 166, 78]:

- *Principio de bivalencia*. Todo enunciado es verdadero o falso, es decir, solo admite dos valores de verdad.
- *Principio de identidad*. Un enunciado es idéntico a sí mismo, es decir, si un enunciado es verdadero, entonces es verdadero.
- *Principio de no contradicción*. Ningún enunciado puede ser tanto verdadero como falso.
- *Principio del tercero excluido*. Un enunciado debe ser verdadero o su negación debe ser verdadera.
- *Declarativa*. La lógica solo se ocupa de proposiciones declarativas, es decir, enunciados que pueden ser verdaderos o falsos. Esta idea está relacionada con el concepto de *apofántico* al que se refiere Aristóteles en sus tratados de lógica, que alude a que la lógica trata con enunciados que afirman o niegan acerca del mundo [58].
- *Funcional de los valores de verdad*. El valor de verdad de un enunciado compuesto por otros enunciados depende del valor de verdad de los enunciados que lo conforman.
- *Asertórica*. No se tiene en cuenta ningún contexto al determinar el valor de verdad de un enunciado.

Entre las lógicas clásicas destacan la lógica proposicional, la lógica de primer orden o de predicados y la lógica de segundo orden [62]. Aunque estas lógicas servían para razonar sobre aspectos "estables" y fijos del mundo real, pronto se hicieron patente sus limitaciones al no contemplar todos los enunciados del razonamiento humano. La veracidad de frases como "Es poco probable que mañana nieve en Carmona", donde se introduce el concepto de probabilidad; "Lebrija está cerca de Sanlúcar de Barrameda", que admite valores de verdad graduales; o "Aquí está el árbol de Navidad del Rockefeller Center", cuya verdad depende del espacio y del tiempo; no puede analizarse a través de la lógica clásica. Aparecieron, ya desde principios del siglo XX, diferentes lógicas que se agruparon bajo el término *lógicas no clásicas*, y que hacen referencia a cualquier sistema formal diferente del clásico, negando algunos de sus principios o introduciendo nuevas condiciones.

La filósofa inglesa Susan Haack establece una distinción entre las lógicas no clásicas, ampliamente aceptada, en [78]. Por un lado, las *lógicas extendidas* que, aunque aceptan que la lógica clásica es pertinente para el análisis de razonamientos, consideran necesario extender el sistema formal de las lógicas clásicas para poder abordar razonamientos informales que previamente no podían ser analizados mediante las herramientas del sistema. Para ello, se introducen nuevos operadores, como temporales o modales, o se trabaja con nuevas reglas y axiomas. Destacan las teorías de la lógica modal [41], la lógica temporal [156] y la lógica probabilística [27]. Por otro lado, *las lógicas divergentes o restringidas* que, aunque utilizan el mismo lenguaje que las lógicas clásicas, consideran que alguno de sus puntos son erróneos, por lo que algunas de sus reglas y principios se modifican. Por ejemplo, destacan la lógica intuicionista [86], la lógica paraconsistente [95] y la lógica difusa [218]. Esta última se estudiará con mayor detalle en la Sección 1.1.2.

Volviendo a la idea original de la lógica como la ciencia del razona-

miento, uno de sus principales objetivos es determinar qué define la validez de un argumento. Un argumento válido es una cadena de enunciados, comenzando con las premisas y terminando con la conclusión, en el que la verdad de las primeras implica la de la segunda. En este caso, en un razonamiento válido, se dice que la conclusión es *consecuencia lógica* de las premisas. Por ejemplo, si consideramos el razonamiento

"Todos los que persiguen obsesivamente algo terminan destruidos"
"El capitán Ahab se obsesiona con Moby Dick"
"El capitán Ahab es destruido"

está claro que, si suponemos que las dos primeras afirmaciones son verdaderas, entonces la tercera se deduce como consecuencia de las dos primeras, siendo, por lo tanto, un caso de razonamiento válido. Esto también ocurre con el razonamiento

"Todos los héroes buscan gloria eterna"
"Ulises es un héroe"
"Ulises busca2 gloria eterna"

Con el propósito de determinar qué condiciones hacen que el tercer enunciado sea consecuencia lógica de los dos anteriores, fijémonos que no parece importar que estemos hablando sobre balleneros obsesionados o héroes de la mitología griega. En general, el siguiente razonamiento también sería válido, aunque no sepamos el contexto específico.

"Todo $\oplus$ tiene la propiedad $\boxtimes$"
"$\dot{+}$ es un $\oplus$"
"$\dot{+}$ tiene la propiedad $\boxtimes$"

Todos estos casos son ejemplos de razonamientos deductivos basados en la regla de inferencia llamada *modus ponens*. Por tanto, como no es necesario comprender el significado de las oraciones, es decir, su parte semántica,

una máquina podría realizar este razonamiento automáticamente fijándose únicamente en la estructura sintáctica de las oraciones y en las reglas lógicas que permiten deducir las conclusiones, en este caso, el *modus ponens*. Esta reflexión es el punto de partida de la teoría de la *deducción automática*, o *razonamiento automático*, como también es conocida, que ha sido fundamental para el desarrollo tanto de la inteligencia artificial como de la programación lógica, concepto de gran relevancia dentro del presente trabajo.

Además, esta teoría se enmarca dentro del área de la lógica conocida como *lógica computacional*, cuyo objetivo es el uso de la lógica formal en los ordenadores, aplicándose en la resolución de problemas en áreas como la teoría de la complejidad computacional; en sistemas de bases de datos; la teoría de lenguajes de programación; la representación del conocimiento; el procesamiento de información de manera eficiente y coherente, y el soporte a sistemas de toma de decisiones; la modelización y automatización de razonamientos lógicos donde se desee un comportamiento inteligente; la inferencia de nuevos conocimientos como consecuencia de un conjunto de hechos; la verificación y corrección de programas; la demostración automática de teoremas; etc. [65].

1.1.2. Lógica difusa

Como hemos comentado, las lógicas no clásicas nacieron para dar respuesta a las limitaciones de la lógica clásica. En el caso de las lógicas divergentes que menciona Susan Haack, se elimina o modifica alguno de los principios que estructuran los razonamientos de la lógica clásica. Por ejemplo, aunque existen enunciados que solo admiten dos grados de verdad, conforme al principio de bivalencia, como "El dramaturgo español Lope de Vega escribió *La discreta enamorada*", la mayoría de los enunciados presentan un cierto grado de incertidumbre y vaguedad. Con respecto a

este último punto, podemos hacer referencia a la *Paradoja de Sorites* [201], atribuida al filósofo griego Eubúlides de Mileto, y que consiste en plantearse en qué momento un montón de arena ya no lo es si se van quitando granos de arena paulatinamente. Un grano de arena es tan insignificante que, aunque lo quitemos, el montón de arena no va a perder su condición como tal. Podemos continuar procediendo de esta forma, quitando granos de arena, hasta llegar a un momento donde el montón de arena pasará a ser un único grano que, obviamente, no es un montón de arena. En algún momento se ha tenido que producir el cambio, aunque no está claro cuándo ha sido. Esta incertidumbre que existe en la determinación de un montón de arena, también se asocia a otros enunciados como "ser calvo", "ser alto", "que una película sea interesante", "que un conjunto de árboles conformen un bosque", "que haga calor", etc.

Para solucionar este inconveniente, en 1965, el matemático azerí Lotfi A. Zadeh introdujo el primer paso hacia la *lógica difusa* en el artículo *Fuzzy Sets*, de la revista *Information and Control* [218]. Zadeh define en este artículo la noción de conjunto difuso, base de la lógica difusa, como una extensión de los conjuntos clásicos a los que le asocia una función de pertenencia, que le asigna a cada elemento del conjunto un número del intervalo unidad en función del grado de pertenencia de dicho elemento al conjunto en particular. Cuanto más cercano sea el valor de la función de pertenencia a uno, mayor será el grado de pertenencia de dicho elemento al conjunto. De esta forma, el uso de las funciones de pertenencia permite representar mejor la incertidumbre que presentan determinados enunciados, como los mencionados anteriormente. Asimismo, en este artículo, Zadeh adapta al marco difuso las definiciones de inclusión de conjuntos y relación, así como las operaciones que se pueden realizar entre conjuntos.

En 1968, Zadeh publicó el artículo *Fuzzy Algorithms* [215], en el que se profundiza en el uso de conjuntos difusos, por ejemplo, en el diseño de algoritmos donde las instrucciones pueden contener términos de la lógica

difusa, como variables lingüísticas, y asignaciones y reglas condicionales *IF-THEN* difusas, dando lugar al concepto de *algoritmo difuso*. Esta aplicación de los conjuntos difusos también supuso uno de los primeros usos de la lógica difusa en los problemas computacionales, demostrando que era posible modelar sistemas complejos con información imprecisa. En particular, Zadeh señala su aplicabilidad en áreas como el procesamiento de información, el control, el reconocimiento de patrones, la identificación de sistemas, la inteligencia artificial y la toma de decisiones a partir de datos incompletos o imprecisos.

En el contexto de la presente tesis, enfocada en el tratamiento de la incertidumbre en las bases de datos, cabe destacar que, en 1973, Zadeh publicó el artículo *Outline of a new approach to the analysis of complex systems and decision processes* [219], donde se resalta la importancia de la lógica difusa para el tratamiento del control, de los sistemas de decisión y de las bases de datos. En concreto, el lógico azerí señala que la lógica difusa juega un papel fundamental en la selección, síntesis y extracción de información relevante de las grandes colecciones de datos, siendo este uno de los aspectos más importantes del pensamiento humano. Zadeh también señala que los enfoques tradicionales de análisis de sistemas no han sido capaces de tratar con los sistemas actuales, dada la falta de comprensión y tratamiento de la ambigüedad en el pensamiento, el comportamiento y el razonamiento humano. Es por ello, por lo que se necesitan nuevas aproximaciones para analizar estos sistemas que puedan admitir el tratamiento de la incertidumbre.

Tal Zadeh en [218, 219], donde destaca una amplia gama de aplicaciones de los algoritmos difusos y, por tanto, de la lógica difusa, que incluye campos como la lingüística, la recuperación de información, la psicología, la economía, la clasificación de patrones, la medicina, la inteligencia artificial, el procesamiento de información, la biología o la taxonomía; la lógica difusa ha demostrado ser eficazmente utilizada en diversos ámbi-

tos. La primera aplicación práctica para el control difuso fue presentada en 1974 por Ebrahim Mamdani para el control de un aparato de vapor [120]. Unos años más tarde, en 1978, la lógica difusa fue utilizada para el control de un horno de cemento por la sociedad danesa F. L. Smidth, constituyendo la primera implementación de la lógica difusa en el ámbito industrial [91]; la empresa japonesa Fuji Electric, dedicada a la fabricación de productos electrónicos y sistemas automatizados, empleó la lógica difusa para el control de un sistema de purificación de agua, mientras que Hitachi, otra empresa del mismo sector, utilizó la lógica difusa para el sistema de control de un tren automático [89]. Actualmente la lógica difusa está siendo usada satisfactoriamente en el diseño y control de electrodomésticos y sistemas de climatización inteligentes [177, 1, 16], en el control del tráfico urbano [35, 140, 12, 169], en sistemas de energías renovables y en aplicaciones fotovoltaicas [186, 161, 30, 8, 123], en la toma de decisiones en los procesos de fabricación [10], en sistemas de control de vehículos eléctricos híbridos [45], en el control de robots [4, 18, 28], en sistemas de sensores militares [52], en finanzas [20, 181], en sistemas de recomendación [144, 107, 126], en seguridad informática [96, 53], en el internet de las cosas [3], en minería de datos [173], en el mundo de las inversiones inmobiliarias [69], en ingeniería [76], en sistemas de riego inteligentes [139], en el área de comunicaciones móviles e inalámbricas [137], en el modelado de procesos mecanizados [133], en la sanidad [141], en el mundo de la cosmética [63], etc.

1.1.3. El problema de la decidibilidad

Uno de los aspectos clave en la lógica computacional es el proceso de deducción automática que, como se señala en [81], consiste en el uso de ordenadores para realizar inferencias lógicas, es decir, para inferir conclusiones válidas a partir de un conjunto de hipótesis y reglas de inferencia,

de manera independiente y autónoma. Para ello, es necesario diseñar algoritmos o procedimientos que permitan decidir, en una secuencia finita de pasos, si una determinada fórmula o enunciado se deduce o no a partir de las hipótesis del sistema. Si en un sistema es posible encontrar tal algoritmo, entonces se dice que satisface la propiedad llamada *decidibilidad*. En el contexto de la deducción automática, esta propiedad no solo marca los límites de los sistemas que pueden ser resueltos a través de algoritmos, sino que determina el éxito y la eficiencia de estos sistemas. Si un sistema es deducible, es decir, satisface la decidibilidad, entonces se llama *sistema automático de decisión* y, en este caso, la deducción automática ofrece resultados fiables y efectivos. Por el contrario, si no es deducible, los sistemas de deducción automática pueden ser incompletos o no terminar para determinados problemas.

En el Congreso Internacional de Matemáticas de 1928, que se celebró en Bolonia, el alemán David Hilbert presentó algunas cuestiones que debían abordarse sobre la concepción de las matemáticas como un sistema formal. En primer lugar, si las matemáticas son completas, en el sentido de si cada afirmación puede ser probada o refutada. En segundo lugar, si las matemáticas son consistentes, es decir, que pasos válidos en una demostración conducen a conclusiones válidas. Finalmente, si las matemáticas son decidibles, es decir, como hemos mencionado, si existe un método que nos permita decidir si cualquier afirmación es cierta dentro del sistema [90]. Estas cuestiones ya comenzaron a ser planteadas por el mismo Hilbert en el Congreso Internacional de París de 1900, donde estableció, en referencia a los axiomas de la aritmética y su consideración como un sistema formal, que era indispensable demostrar que un número definido de pasos lógicos basados en dichos axiomas no pueden conducir nunca a resultados contradictorios [88]. En 1931, el matemático austriaco Kurt Gödel presentó sus famosos teoremas de incompletitud en el artículo *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Syste-*

me I [73]. El Primer Teorema de Incompletitud señala que, en un sistema formal basado en la aritmética, existe al menos una fórmula para la cual ni la fórmula ni su negación pueden ser demostradas dentro del sistema, siendo dicho sistema incompleto y, por lo general, no podemos encontrar un algoritmo que determine si una fórmula o su negación pueden deducirse de las hipótesis del sistema, es decir, es no decidible. Por otro lado, el Segundo Teorema de Incompletitud establece que la consistencia del sistema no puede ser probada dentro del sistema mismo, demostrando una limitación. Por tanto, con estos teoremas, Gödel mostró que no todo sistema formal es completo, y, en consecuencia, mostró que existen fórmulas de las que no se pueden deducir su veracidad o falsedad algorítmicamente, es decir, no son decidibles.

El artículo anterior permitió comenzar a verificar que existen sistemas formales que no son decidibles. Esta circunstancia se aclaró ostensiblemente en 1936 cuando el matemático y lógico inglés Alan Turing publicó el artículo *On Computable Numbers, with an Application to the Entscheidungsproblem* [194]. El concepto "*Entscheidungsproblem*", también conocido como "*Hilbert's Entscheidungsproblem*", traducido del alemán como *Problema de Decisión*, fue planteado por primera vez en 1928 por David Hilbert y Wilhelm Ackermann, en el libro *Grundzüge der theoretischen Logik* [87]. Estos autores lo definieron como "el problema de decidir si una fórmula dada es universalmente válida; [...] decidir mediante un método eficaz y finito si una expresión lógica dada es lógicamente verdadera o no" [87], donde, en el contexto de la lógica de predicados, según el Teorema de Completitud de Gödel [83], las fórmulas universalmente válidas son aquellas que se pueden deducir de los axiomas del sistema. Por tanto, se trata de encontrar un algoritmo o procedimiento general para decidir si una determinada fórmula puede deducirse a partir de los axiomas de un sistema formal.

En 1936, el ya mencionado Alan Turing demostró, en [194], que no exis-

te ningún método general para determinar si una fórmula dada de lógica de primer orden puede ser probada a partir de las hipótesis del sistema, constituyendo una prueba de que el problema de decisión de Hilbert no tenía solución. De manera independiente y en el mismo año, el lógico estadounidense Alonzo Church también llegó a la misma conclusión [36]. En consecuencia, los trabajos de Gödel, Turing y Church habían demostrado que no todos los sistemas formales son decidibles y, en particular, la lógica clásica de predicados no es decidible. Por tanto, al contrario de lo que sucede con la lógica proposicional, en la lógica de predicados no existe un algoritmo general que sea capaz de decidir si cada fórmula dada se puede demostrar a partir de los axiomas del sistema. En estos casos, la deducción automática aspira a encontrar un algoritmo funcional que sea capaz de demostrar la veracidad de algunas fórmulas en un número finito de pasos, lo que se conoce como el *problema de la semidecisión*.

Por lo tanto, al no disponer, en la mayor parte de los casos, de un algoritmo general de decisión, se pueden considerar diversas alternativas para definir la tarea concreta de un sistema de deducción automática. Por un lado, verificar si las conclusiones se pueden deducir a partir de las hipótesis del sistema, en cuyo caso, es recomendable ofrecer una demostración automáticamente (abducción). Por otro lado, formular nuevas proposiciones derivadas de las hipótesis (inducción).

1.1.4. Componentes de una lógica computacional

Para conseguir los objetivos citados anteriormente con respecto a los problemas de semidecisión, una lógica computacional necesita cuatro componentes, a saber, una sintaxis, una semántica, una teoría de la demostración, siendo estas componentes comunes a cualquier lógica formal, y un algoritmo de decisión.

La sintaxis se encarga de determinar, a través de una serie de reglas y

estructuras, cómo deben formarse correctamente las expresiones y construcciones lógicas para representar formalmente el lenguaje, independientemente de que esas combinaciones de símbolos sean verdaderas o falsas. La sintaxis se fundamenta en la noción de las fórmulas bien formadas aceptadas por la gramática de un lenguaje formal.

Por otro lado, la semántica se encarga de dar significado a las fórmulas atendiendo a su estructura sintáctica. A partir de la "interpretación" de las fórmulas y expresiones lógicas en un contexto determinado, se pueden inferir o deducir nuevos conocimientos. En consecuencia, la semántica se erige como un concepto fundamental para establecer las nociones de *inferencia* y *deducción*. La principal definición dentro de la semántica de un lenguaje es la de *modelo*, que es una forma de interpretar semánticamente las proposiciones del lenguaje, estableciendo relaciones entre los elementos del modelo y las fórmulas del lenguaje, con el objetivo de establecer su significado. A partir de la noción de modelo, se pueden introducir las definiciones de satisfacilidad, validez e inferencia semántica, siendo esta última un proceso por el que se deduce una conclusión a partir de unos hechos iniciales que son las premisas, atendiendo al significado de las fórmulas. En este caso, se escribe $\Gamma \vDash A$, y se lee "Γ implica semánticamente A" o "A es consecuencia semántica de Γ."

La teoría de la demostración está formada por un conjunto de reglas de inferencia que permiten deducir y probar fórmulas (*teoremas*) a partir de otras, las premisas o hipótesis, a través, exclusivamente, de la estructura sintáctica de las expresiones lógicas, es decir, se atiende únicamente a la sintaxis y no a la semántica. Esta delimitación a la estructura sintáctica representa uno de los principales aspectos que hacen posible la automatización de todo el proceso de deducción, como ya mencionamos al finalizar la Sección 1.1.1. Como las reglas de inferencia posibilitan la derivación de fórmulas a través de su estructura sintáctica, podemos introducir el concepto de *consecuencia sintáctica*, $\Gamma \vdash A$, que significa que A se puede dedu-

cir a partir de Γ aplicando sucesivamente las reglas de inferencia.

Con respecto a la relación entre los conceptos de consecuencia semántica y consecuencia sintáctica, existen dos cuestiones fundamentales que son siempre deseables en toda teoría de la demostración. Por un lado, que el proceso sea *correcto*, es decir, que todo lo demostrable, desde un punto de vista sintáctico, sea válido, desde un punto de vista semántico, siendo una de las propiedades que, según el lógico inglés Robinson, todo principio de inferencia debe tener [162]. Dicho de otra forma, si $\Gamma \vdash A$, entonces $\Gamma \vDash A$. Por otro lado, que sea *completo*, es decir, que todo lo válido, desde un punto de vista semántico, sea demostrable, desde un punto de vista sintáctico. En otras palabras, si $\Gamma \vDash A$, entonces $\Gamma \vdash A$; lo cual también es deseable, aunque esto, como ya argumentó Gödel, no siempre se va a poder asegurar, incluso para lógicas muy potentes y flexibles.

Finalmente, la última componente de un sistema de lógica computacional es un algoritmo de decisión, encargado de establecer sintácticamente aquellas deducciones que están permitidas, dando, además, un procedimiento de decisión para determinar, en una secuencia finita de pasos, si una fórmula dada es válida o no. Mientras que las tres primeras componentes de la lógica computacional están relacionadas con la parte de la lógica a la que hace referencia Robert Kowalski en su distinción entre lógica y control en [112], el algoritmo de decisión se asocia con la parte del control, es decir, representa la parte procedimental que se encarga de utilizar la lógica, en concreto seleccionando las reglas de inferencia, para resolver un determinado problema.

1.1.5. Deducción automática

Las primeras nociones de programación lógica surgieron a principios de los años 1970 como resultado directo de los avances en inteligencia artificial y demostración automática de teoremas, también conocido este

último, en determinados contextos, como deducción automática. En concreto, los investigadores de principios de la segunda mitad del siglo XX buscaban crear programas declarativos que pudieran resolver problemas aplicando los principios de la lógica, en los cuales el programador, es decir, la persona que escribe el programa, define los hechos y las reglas lógicas, dejando que el sistema deduzca las conclusiones sin necesidad de especificar todo el proceso a realizar. Los avances en el terreno de la deducción automática ayudaron a este fin.

Uno de los primeros artículos clave en el área de la deducción automática, que inspiró posteriormente el trabajo de los investigadores iniciales en programación lógica, fue *Investigations in proof theory*, del lógico francés Jacques Herbrand, de 1930 [84]. Este artículo es fundamental en la teoría de la demostración matemática, aportando el famoso Teorema de Herbrand, esencial en la atomatización del proceso de deducción. La idea central de este teorema consiste en transformar fórmulas complejas abstractas de lógica de primer orden en otras fórmulas donde las variables están instanciadas, es decir, se trata de fórmulas de primer orden con casos específicos (sin variables). Tomando como base este planteamiento, el teorema ofrece un procedimiento que permite demostrar fórmulas complejas a partir de la veracidad de instancias particulares. Este procedimiento no solo permitió abordar el estudio de la veracidad de fórmulas complejas, sino que, gracias al proceso de descomponer fórmulas complejas de lógica de primer orden en conjuntos de proposiciones más simples, fue posible acometer la resolución de problemas lógicos más complejos con mayor efectividad.

A partir de la década de 1960, diferentes autores desarrollaron computacionalmente las ideas introducidas por Herbrand en los años treinta, con el objetivo de automatizar el razonamiento lógico, en particular, en el área de la demostración automática de teoremas. Entre estos trabajos destacan *An improved proof procedure*, de Dag Prawitz [155]; *A proof method for quan-*

tification theory, de P. Gilmore [68]; *A computing procedure for quantification theory*, de Martin Davis y Hilary Putnam [46]; *Toward Mechanical Mathematics*, de Hao Wang [198], todos publicados en 1960. Las contribuciones de los distintos autores alcanzaron su máxima expresión en 1965 con la aportación del lógico inglés John Alan Robinson, *A machine-oriented logic based on the resolution principle* [162].

Este artículo aporta ideas fundamentales para el desarrollo de la deducción automática, apoyándose en el Teorema de Herbrand. En primer lugar, Robinson señala que, para el proceso de deducción automática, un principio de inferencia únicamente necesita verificar dos propiedades importantes. La primera de ellas es la corrección, *soundness* en su notación inglesa, de la que ya hemos hablado en la Sección 1.1.4. Esta propiedad hace referencia a que el principio de inferencia, si las premisas son verdaderas, no puede conducir a conclusiones falsas; como dice Robinson, "únicamente se permite deducir consecuencias lógicas a partir de las premisas" [162]. La corrección proporciona la seguridad de saber que la conclusión que la máquina obtiene al aplicar el procedimiento es verdadera y confiable. La segunda propiedad de los principios de inferencia a la que hace referencia Robinson es la efectividad, *effectiveness* en inglés, estableciendo que "debe ser algorítmicamente decidible si una supuesta aplicación del principio de inferencia es realmente una aplicación del mismo" [162], es decir, que debe existir un procedimiento bien definido, con las reglas bien claras, que permitan decidir mecánicamente si el principio de inferencia ha sido aplicado correctamente, sin necesidad de contar con la intervención del ser humano para determinar si un paso en concreto es correcto o si es necesario interpretar algo.

No dejándose influir por la simplicidad de los principios de inferencia empleados hasta ese momento en el ámbito de la demostración automática de teoremas, en los que todos los pasos deductivos que se dan en un proceso de razonamiento, es decir, para pasar de un argumento al siguien-

te, tienen que ser lo bastante sencillos como para que el intelecto humano los pueda entender sin ninguna duda o ambigüedad, Robinson introduce un principio de inferencia más complejo, que no supone ninguna desventaja con respecto a la falta de comprensión del proceso ya que quienes se van a encargar de llevar a cabo este principio de inferencia son las "modernas máquinas de computación" de los años 1960, que son máquinas computacionales cada vez más potentes.

Específicamente, Robinson introduce en [162] un principio de inferencia llamado *principio de resolución*, que es una extensión del *modus ponens* clásico. Fue diseñado para que las máquinas pudieran usar un procedimiento mecánico que permitiera obtener conclusiones lógicamente válidas y correctas, atendiendo a la noción de corrección planteada anteriormente, y que pudiera ser ejecutado de manera eficaz por un algoritmo, evitando, de esta forma, la necesidad de intervención humana. Este principio de inferencia se basa, además, en el concepto de unificación entre variables en el contexto de la lógica de predicados, que Robinson utiliza en este artículo, habiendo sido introducido previamente por Herbrand en [84]. Se trata de un método para encontrar una sustitución que haga coincidir dos variables, siendo muy importante para el proceso de deducción automática porque permite a las máquinas entender cómo dos variables pueden ser compatibles o tener el mismo significado. El principio de resolución fue fundamental para automatizar los procesos de deducciones lógicas, dando un paso hacia el desarrollo de la programación lógica.

1.1.6. Programación lógica

Estos dos conceptos, el principio de resolución y el método de unificación, perfeccionados por Robinson como métodos algorítmicos para que las máquinas pudieran realizar inferencias de manera automática en lógica de primer orden [162], junto con los trabajos iniciales de Herbrand [84],

las ideas de Russell y North Whitehead [200] y todos los lógicos que pusieron su granito de arena, sentaron definitivamente los cimientos de la programación lógica a principios de los años 1970.

La programación lógica se fundamenta en la idea de que la lógica simbólica es un lenguaje apropiado para expresar y manejar el conocimiento, es decir, su objetivo es crear un lenguaje de programación basado en la lógica. A este fin ayudaron también todos los trabajos anteriores sobre deducción automática [162, 155, 68, 46, 198], que contribuyeron a la concepción de la deducción como una forma de computación, junto con la visión de que la lógica tenía una interpretación procedimental.

Desde la aparición de los primeros ordenadores modernos a partir de los años 1940, basados principalmente en los conceptos desarrollados por John von Neumann, la mayoría han utilizado lenguajes para expresar problemas y plantear soluciones que han sido concebidos desde una perspectiva bastante compleja, desde el enfoque de la ingeniería de la computación únicamente, es decir, había estado pensado para el ordenador y no para la comprensión del ser humano. Como mencionan los investigadores Leon Sterling y Ehud Shapiro en [184] en relación con la búsqueda de nuevos lenguajes de programación, "el impedimento dejó de ser la incapacidad del ordenador para realizar las instrucciones del ser humano, y pasó a ser la incapacidad del ser humano para instruir, o programar, el ordenador. Comenzó la búsqueda de lenguajes de programación convenientes para el uso humano". Por tanto, los ordenadores se han programado tradicionalmente utilizando lenguajes complejos y bastante alejados del lenguaje natural del ser humano, siendo complicados de comprender. En cambio, un lenguaje de programación basado en la lógica matemática facilita mucho el proceso de comprensión gracias a la función principal de la lógica, que es estructurar los esquemas válidos de razonamiento, facilitando, por consiguiente, la comprensión de los programas para el intelecto humano. Además, la programación lógica permite abordar problemas que

hasta entonces no se podían con la programación tradicional, como la especificación de problemas complejos, la gestión de la incertidumbre o la deducción automática.

A finales de la década de 1960 y comienzos de la de 1970 se publicaron varios artículos centrados en la creación de un lenguaje de programación basado en la lógica (de primer orden), entre los cuales destacan [75, 82] y, especialmente, [37, 111]. Todo ello supuso una idea revolucionaria puesto que, hasta el momento, la lógica solo se había utilizado como lenguaje de especificación, pero nunca en programación. Esto último fue posible gracias a la interpretación procedimental de la lógica, permitiendo que la especificación lógica del problema pudiera ser ejecutada por la máquina, aspecto que fue probado en [111].

Concretamente, los equipos de investigación del informático Alain Colmearauer, de la Universidad de Aix-Marsella, y del lógico Robert Kowalski, de la Universidad de Edimburgo, colaboraron en el desarrollo de lo que se convirtió en el lenguaje de programación `ProLog`, acrónimo de *PROgrammation en LOGique*. El primero estaba centrado en técnicas de procesamiento del lenguaje natural, mientras que el segundo introdujo una extensión del principio de resolución de Robinson adaptada al contexto de la programación lógica, llamada resolución SL, de *Linear resolution with Selection function* [113].

Esta exitosa fusión entre el procesamiento del lenguaje natural y la demostración automática de teoremas y, por extensión, la deducción automática, empezó a dar sus frutos a partir de 1972 cuando el primer intérprete de `ProLog` fue implementado por Philippe Roussel, del equipo de Colmerauer. Su primera aplicación, así como la primera mención en un artículo, ocurrieron cuando se planteó un sistema de comunicación entre el hombre y la máquina en *Un système de communication homme-machine en français* [37], en 1973, de Alain Colmerauer, Henry Kanoui, Philippe Roussel y Robert Pasero. Los programas implementados en lenguaje `ProLog`,

que es declarativo, se diseñaron para realizar deducciones automáticas; donde se definen hechos y reglas, para después formular consultas, sin especificar cómo el motor lógico debe inferir la solución.

Aunque esta primera implementación de `ProLog` utilizó como método de inferencia la resolución SL introducida en [113], posteriormente, en 1974, Kowalski presentó una variante de la resolución SL en el artículo *Predicate Logic as Programming Language* [111], llamada *resolución SLD*, que viene de *Selective Linear resolution with Definite clauses*, es decir, *resolución Lineal Selectiva con cláusulas Definidas*. Este nuevo principio de inferencia se sustenta en la interpretación procedimental de las cláusulas de Horn, que facilita su ejecución por una máquina. Además, en este artículo y, posteriormente, en *Algorithm = Logic + Control* [114] y *Logic for Problem Solving* [112], ambos publicados en 1979, Kowalski establece los fundamentos teóricos de la utilización de la lógica de predicados como lenguaje de programación, que se basa, como hemos mencionado, en la interpretación procedimental de especificaciones lógicas como "B if A_1 and $\ldots$ and A_n" (que son implicaciones $p \leftarrow q$ que se ponen como cláusulas de Horn, es decir, $\neg p \vee q$).

En estos dos artículos, como parte del marco teórico, Kowalski plantea una distinción entre dos aspectos fundamentales de los algoritmos, la *lógica* y el *control*, que expresa como

$$\texttt{Algoritmo} = \texttt{Lógica} + \texttt{Control}$$

Esta distinción es la base del uso de la lógica como lenguaje de programación. Por un lado, como se menciona en [118], "*The logic is the statement of what the problem is that has to be solved*", esto es, que la lógica especifica el problema a resolver, mientras que, por otro lado, "*The control is the statement of how it is to be solved*" [118], es decir, que el control precisa cómo se tiene que resolver. Kowalski plantea en [112] que la lógica se ocupe del *qué* (lo que el algoritmo hace), mientras que el control del *cómo* (la manera en

que el algoritmo lo hace). También añade que, a diferencia de la programación tradicional, en la programación lógica el programador, es decir, la persona que escribe el programa lógico, únicamente necesita especificar la parte lógica de los algoritmos. La parte de control será ejercida por el ejecutor del programa, esto es, la componente que se encarga de la ejecución real del programa lógico, ya sea siguiendo sus propias decisiones de manera autónoma o instrucciones de control proporcionadas por el programador. Esta distinción supone una gran ventaja puesto que el programador debe centrarse exclusivamente en especificar la componente lógica del proceso. Todo esto también permite que los algoritmos se puedan perfeccionar sistemáticamente a través de la mejora de la parte de control, sin modificar la componente lógica.

Tras haber visto la distinción propuesta por Kowalski, veamos cada una de las componentes por separado. Con respecto a la componente de control, como hemos comentado, el principal método de inferencia que se utilizó en programación lógica ha sido la ya mencionada resolución SLD.

Por otro lado, en la parte de la lógica, la noción fundamental es la de cláusula de Horn, descrita en 1951 por Alfred Horn [93]. Las cláusulas de Horn son un caso particular de cláusulas en las que se consideran la disyunción de un literal positivo, como mucho, y una cantidad ilimitada de literales negativos, es decir, $A \vee \neg B_1 \vee \ldots \vee \neg B_n$, donde $A, B_1, \ldots, B_n$ son literales, con $n \geq 0$. Al contar con un único literal positivo, el proceso de inferencia se simplifica, lo que lo hace ideal para que el proceso de resolución sea computacionalmente eficiente. De hecho, como hemos visto, la fundamentación teórica que realiza Kowalski de la programación lógica se basa en la interpretación procedimental de estas cláusulas.

En relación con la implementación de `ProLog`, Colmerauer y Roussel mencionan en [38] que "se tomó una decisión draconiana: a costa de la incompletitud, optamos por una resolución lineal con unificación solo entre las cabezas de las cláusulas. Sin saberlo, habíamos descubierto la estra-

tegia que es completa cuando solo se utilizan cláusulas de Horn. Robert Kowalski demostró este punto más tarde en [111]". Por tanto, con el principio de resolución SL que utilizaron inicialmente, el procedimiento no era completo, no pudiendo derivar todas las conclusiones válidas en casos donde las premisas incluían cláusulas generales. En cambio, con la resolución SLD restringida a cláusulas de Horn la estrategia sí era completa, lo que supuso una gran ventaja. Asimismo, las cláusulas de Horn son la base de la sintaxis en la programación lógica a través de la expresión lógica equivalente $A \leftarrow B_1 \wedge \ldots \wedge B_n$, simplificado como $A \leftarrow B$; permitiendo una representación clara de las reglas y los hechos de un programa lógico. En este caso, A es la cabeza de la regla, es decir, la conclusión, mientras que B conforma el cuerpo de la regla, especificando las condiciones que deben cumplirse para que A sea verdadero.

Posteriormente, en 1976, Kowalski, junto al holandés-canadiense Maarten van Emden desarrollaron una semántica formal para el lenguaje de los programas lógicos en *The semantics of predicate logic as a programming language* [61]. En particular, en el contexto de lógica de primer orden, los autores concluyen que la semántica operacional, que describe cómo se ejecutan los programas o sistemas lógicos paso a paso, forma parte de la sintaxis relacionada con la teoría de la demostración (concepción sintáctica), mientras que la semántica de punto fijo es un caso específico de la semántica teórica de modelos (concepción semántica). Además, la "equivalencia" entre la semántica operacional y la semántica de punto fijo se considera un caso particular del Teorema de Completitud de Gödel, que establece que todo enunciado lógicamente válido puede ser probado dentro de un sistema formal y, por tanto, la validez de los enunciados se puede establecer tanto sintácticamente como semánticamente.

Las primeras generalizaciones y extensiones de la programación lógica, a la que nos referiremos con el apelativo de *clásica*, al ser la primera que se estudió, comenzaron a aparecer a partir de finales de la década de 1970

y principios de los años 1980, con el objetivo de abordar las limitaciones existentes en el caso clásico, a la vez que se buscaba mejorar la expresividad y la aplicabilidad de los programas lógicos.

Entre estas generalizaciones destacan: la *programación lógica con restricciones* [94], extensión del caso clásico que permite trabajar con restricciones, tanto en las reglas lógicas como en las soluciones obtenidas; la *programación lógica inductiva* [138], tratándose de una fusión entre la programación lógica y el aprendizaje inductivo, que permite que el sistema aprenda reglas e hipótesis a partir de ejemplos concretos y, por tanto, que nuevos conocimientos se induzcan a partir de los datos observados; la *programación lógica abductiva* [105], que es una extensión del caso clásico que permite trabajar problemas que parten de observaciones y buscan las posibles hipótesis que puedan dar lugar a esas observaciones, seleccionándose posteriormente, como solución del problema, aquellos supuestos que sean más plausibles; la *programación lógica con anotaciones* [185], que es una extensión de la programación lógica utilizando anotaciones de dos tipos; una sobre la certeza y otra sobre la información de cada una de las fórmulas consideradas en cada regla. La *programación lógica probabilista* [170], donde el conjunto de verdad *booleano* de la programación lógica clásica se extiende al intervalo unidad, y se asignan valores probabilísticos a los hechos y reglas, permitiendo modelar problemas donde los hechos presentan incertidumbre; la *programación lógica no monótona* [66], donde se trabaja con lógicas con operadores de negación, la *programación lógica concurrente* [175], donde se incluye la concurrencia en la programación lógica, permitiendo que varios procesos se ejecuten simultáneamente. Finalmente, nos encontramos dos paradigmas de programación que serán fundamentales para el presente trabajo, uno extensión del otro, y que se abordarán de manera más detallada en la próxima sección. Por un lado, la *programación lógica difusa* [196], que es una extensión de la programación lógica utilizando la teoría de conjuntos difusos y que permite manejar la incertidumbre en el razonamiento

lógico. Por otro lado, la *programación lógica multiadjunta* [124], que generaliza el caso anterior considerando estructuras algebraicas más generales e incorporando, por tanto, una mayor flexibilidad para el tratamiento de datos.

1.1.7. Programación lógica multiadjunta

En las secciones anteriores hemos abordado el origen de la lógica tradicional y sus limitaciones al considerar un conjunto de valores de verdad bivaluado, que restringe su capacidad para representar situaciones reales, lo que llevó al surgimiento de la lógica difusa. La programación lógica clásica también asimiló e incorporó los nuevos conceptos derivados de la lógica difusa, permitiendo definir reglas y hechos que pudieran reflejar mejor los conocimientos a partir de datos imprecisos o incompletos. Todo esto dio lugar a una extensión de la programación lógica, la *programación lógica difusa* [196].

Aunque con este nuevo paradigma de programación lógica se había logrado un sistema lógico mucho más asequible para modelizar el razonamiento con la lógica difusa, seguía teniendo diversas limitaciones. Por ejemplo, se utiliza un solo tipo de implicación para representar las reglas difusas, lo cual limita considerablemente la adaptabilidad de estos sistemas a diferentes entornos. Por esta razón, en 2001, los investigadores Jesús Medina, Manuel Ojeda-Aciego y Peter Vojtáš plantearon un nuevo paradigma de programación lógica difusa, llamada *programación lógica multiadjunta*, en el artículo *Multi-Adjoint Logic Programming with Continuous Semantics*, que se erige como una generalización tanto de la programación lógica difusa [196] como de la programación lógica monótona y residuada [44] [43], entre otras, y en el que diferentes tipos de conectores lógicos, como implicaciones, conjunciones, disyunciones y agregadores, pueden usarse para la construcción de las reglas de un programa lógico. Esta no

es la única ventaja que presenta el marco multiadjunto, ya que considera restricciones menos estrictas sobre las propiedades que deben cumplir los conectores; por ejemplo, no es necesario el requerimiento de la conmutatividad y la asociatividad. De esta forma, se consigue una mayor expresividad y flexibilidad en la representación de sistemas de conocimiento más generales y de manera más realista.

La construcción matemática que posibilita el uso de diferentes operadores lógicos en las reglas es la Ω-álgebra multiadjunta, basada en el concepto de retículo multiadjunto, y que fue introducida en [124, 125].

Estas estructuras algebraicas dependen, principalmente, del concepto de *par adjunto*, que fue introducido en un contexto lógico en 1979 por el matemático Jan Pavelka, en *On fuzzy logic I, II, III* [146, 147, 148]. Pavelka interpretó la estructura de conjunto parcialmente ordenado asociado al conjunto de valores de verdad, como una categoría, y la relación entre las implicaciones y conjunciones como funtores en esta categoría. Los pares adjuntos no solo generalizan las nociones clásicas de implicación y conjuntor, sino que permiten extender la regla de inferencia del *modus ponens* clásico al entorno difuso a través de la propiedad de adjunción, concepto introducido por el matemático holandés Daniel Kan en el contexto general de la teoría de categorías en 1958 [106]. Esta propiedad, que se corresponde con la adjunción categórica, establece que, dado un conjunto parcialmente ordenado $\langle P, \preceq \rangle$ y $x, y, z \in P$, se cumple $x \preceq (z \leftarrow y)$ si y solo si $(x\&y) \preceq z$. Por un lado, la propiedad de adjunción afirma que la siguiente regla de *modus ponens* difusa es válida: si x es una cota inferior de $\psi \leftarrow \varphi$, e y es una cota inferior de φ, entonces una cota inferior z de ψ es $x\&y$. Por otro lado, asegura que el valor de verdad de $z \leftarrow y$ es el valor maximal x que satisface $x\&y \preceq z$. Por tanto, en el entorno multiadjunto, el hecho de poder utilizar varios pares adjuntos nos permite trabajar con diversas reglas de inferencia *modus ponens* difusas [80].

Desde un punto de vista práctico, la programación lógica difusa se ha

implementado a través de varias extensiones de `ProLog`, que son capaces de manejar conocimiento difuso e incierto. Por ejemplo, Bousi∼Prolog [98, 100], que actualiza la separación que Kowalski realiza entre lógica y control en [114] a una clara distinción entre lógica, conocimiento difuso y control; siendo esta última componente realizada automáticamente por el sistema a través de un procedimiento de resolución llamado *resolución SLD débil*, similar al introducido por Kowalski en [111], pero con un proceso de unificación más flexible. Otro ejemplo a destacar es FASILL [99], acrónimo de *Fuzzy Aggregators and Similarity Into a Logic Language*, que combina un algoritmo de unificación débil, basado en relaciones de similitud, junto con una amplia variedad de conectores y operadores de agregación difusos. Al tratarse de una extensión del sistema FLOPER, *Fuzzy LOgic Programming Environment for Research* [134], FASILL incorpora también la programación lógica multiadjunta. Estas extensiones se han aplicado recientemente en distintas disciplinas, tales como redes neuronales [135], aprendizaje automático explicable [183], toma de decisiones [29], redes sociales [7], descubrimiento de conocimiento [164], computación en la nube [191], técnicas para resolver problemas computacionales SAT (*propositional SATisfiability problem*) y SMT (*Satisfiability Modulo Theories*) [6], clasificación de textos [163], web semántica [5], modelos metabólicos a escala genómica [119].

A continuación, analizamos las partes fundamentales de la programación lógica multiadjunta según las componentes de la lógica computacional comentadas en la Sección 1.1.4.

La sintaxis en la programación lógica multiadjunta está basada en la noción de *programa lógico multiadjunto*, que consiste en un conjunto de reglas lógicas que modelan colecciones de datos, de la forma $\langle C \leftarrow \mathcal{B}; \vartheta \rangle$. Por un lado, C se denomina la *cabeza* de la regla, que es un elemento del lenguaje, que puede ser un símbolo proposicional, como en el caso de la lógica proposicional, o un átomo, si nos estamos refiriendo a la lógica de predica-

dos. Por otro lado, $\mathcal{B}$, llamado el *cuerpo* de la regla, es una fórmula bien formada, es decir, sintácticamente válida, formada a partir de elementos del lenguaje más sencillos, como átomos o símbolos proposicionales, mediante el uso de conjunciones $\&_1, \ldots, \&_r$ y $\wedge_1, \ldots, \wedge_l$, disyunciones $\vee_1, \ldots, \vee_k$ y/o agregadores $@_1, \ldots, @_m$. Por último, ϑ es un valor de verdad, que se puede obtener a través, tanto de la observación, como de la experimentación a partir de (grandes) cantidades de datos. En el caso multiadjunto, que como ya hemos comentado, generaliza el caso difuso, el valor de verdad ϑ pertenece a un retículo. En este caso, en la regla lógica $\langle C \leftarrow \mathcal{B}; \vartheta \rangle$ se dice que C está implicado por $\mathcal{B}$ con un grado de verdad ϑ. Estas reglas se completan con observaciones particulares en la forma de *hechos*, que son reglas cuyo cuerpo es $\top$ y representan los axiomas y las hipótesis que las reglas de inferencia de nuestro sistema formal utilizarán para el proceso de razonamiento deductivo.

Como ya hemos comentado en secciones anteriores, una de las mayores preocupaciones en la programación lógica es definir un método a partir del cual se puedan determinar aquellas deducciones que son válidas y que se pueden demostrar a partir de las reglas y los hechos definidos. Para ello, se necesita definir una semántica en este marco multiadjunto. Existen diferentes enfoques semánticos, cada uno de los cuales interpreta y otorga significado al programa lógico desde perspectivas específicas.

La semántica declarativa consiste en otorgar significado al programa lógico a través de los modelos que satisfacen las fórmulas del programa [118]. Para poder determinar la veracidad o la falsedad de una regla en un programa lógico multiadjunto es necesario otorgar algún significado a los elementos del lenguaje que aparecen en su construcción. Para ello, se introduce la noción de *interpretación*, siendo una forma de otorgar significado a las variables y elementos básicos del programa. Como una de las propiedades de esta lógica es que es funcional de los valores de verdad, al otorgar significado a los elementos básicos podemos obtener el valor de

verdad de la misma fórmula. Aquellas interpretaciones que hacen que todas las reglas del programa sean verdaderas, es decir, cuando la interpretación de la regla como fórmula lógica es mayor que su grado de verdad, se llaman *modelos*.

En 1976, Maarten van Emden y Robert Kowalski publicaron el artículo *The Semantics of Predicate Logic as a Programming Language* [61], en el que definen una semántica de punto fijo para la programación lógica de predicados, basada en los trabajos anteriores de Dana Scott [172], de 1970, y Jacobus de Bakker [11], de 1971. Siguiendo lo establecido en las semánticas de punto fijo, donde el significado de un procedimiento recursivo se determina como el menor punto fijo de una transformación asociada al procedimiento, y teniendo en cuenta la interpretación procedimental de las cláusulas de Horn, van Emden y Kowalski introducen una transformación $T_{\mathbb{P}}$, asociada al programa lógico, que recibe el nombre de *operador de consecuencias inmediatas*, porque genera, de manera directa, las consecuencias derivadas de un conjunto de hechos dados a partir de un programa lógico. En la semántica de punto fijo, las consecuencias óptimas del programa se obtienen a partir del menor punto fijo de este operador, que es donde el programa alcanza la "estabilidad". Esto es posible gracias a su monotonía, a que se está trabajando con retículos completos y a la aplicación del teorema de punto fijo del lógico polaco Alfred Tarski, que introduce en [189]. Posteriormente, Medina, Ojeda-Aciego y Vojtáš generalizaron tanto el operador de consecuencias inmediatas $T_{\mathbb{P}}$, como la semántica de punto fijo de van Emden y Kowalski al contexto multiadjunto en [124].

Por otro lado, una vez que se ha estudiado una semántica de punto fijo, para el desarrollo de un modelo computacional es necesario especificar una semántica operacional que permita definir un procedimiento de razonamiento automático para obtener deducciones e información del sistema modelado, y para responder a consultas al programa. Esta semántica describe el significado de los programas especificando cómo se ejecutan,

paso a paso. En *The Semantics of Predicate Logic as a Programming Language* [61], asentado asimismo en los trabajos previos mencionados anteriormente [172, 11] , se introduce una semántica operacional para la programación lógica de predicados. Kowalski y van Emden mencionan en este artículo que "definir una semántica operacional para un lenguaje de programación consiste en definir un intérprete independiente de la implementación para él. En el caso de la lógica de predicados, el procedimiento de demostración actúa de manera similar a dicho intérprete" [61]. Dicho procedimiento de demostración está conformado, entre otras cosas, por un conjunto de reglas de inferencia, de manera similar a la teoría de la demostración que mencionamos en la Sección 1.1.4.

Recordemos que, como ya apuntamos en la Sección 1.1.6, van Emden y Kowalski señalan que la semántica operacional forma parte de la sintaxis relacionada con la teoría de la demostración, mientras que la semántica de punto fijo se relaciona con la semántica teórica de modelos. En consecuencia se puede establecer una equivalencia entre ambas semánticas a partir del Teorema de Completitud de Gödel.

En el marco multiadjunto, que es el que nos ocupa, una semántica operacional fue presentada por Medina, Ojeda-Aciego y Vojtáš en *A Procedural Semantics for Multi-Adjoint Logic Programming* [125].

Con respecto a la teoría de la demostración, al tratarse de una parte puramente sintáctica, esta está formada por las reglas admisibles en la semántica operacional, junto con los procesos de unificación y sustitución. Por el contrario, la parte del control relacionada con la existencia de un algoritmo de decisión corresponde a la manera en que se combinan las reglas admisibles para obtener conclusiones.

En el presente trabajo nos centraremos en la lógica proposicional y en la de primer orden. Además, trabajaremos con programas lógicos multiadjuntos definidos, es decir, los operadores en los cuerpos de la reglas serán monótonos crecientes en todos sus argumentos.

1.1.8. Cuantificadores generalizados

Expresiones como "Existe al menos un tenista español que ha ganado el torneo de *Roland Garros*", "Todas las películas de los hermanos Coen son buenas" o "Casi todos los grandes éxitos de *Creedence Clearwater Revival* fueron grabados a finales de la década de 1960" contienen cuantificadores, es decir, expresiones lógicas y matemáticas que indican cantidad, modificando en consecuencia el significado de las frases. Los cuantificadores más básicos y esenciales son el existencial $\exists$ y el universal $\forall$, conocidos como cuantificadores de primer orden. El primero de ellos indica que existe al menos un elemento de un conjunto dado que cumple una determinada propiedad, mientras que el segundo indica que todos los elementos del conjunto satisfacen la propiedad. La cuantificación resulta fundamental, entre otras cosas, para los sistemas de conocimiento y en la toma de decisiones. Por ejemplo, en un diagnóstico médico hay una diferencia considerable entre "La mayoría de los niveles de glucosa están elevados" y "Casi ninguno de los niveles de glucosa están elevados". Por tanto, debido a la importancia de los cuantificadores en la expresividad del lenguaje, no es raro que se estudiarán desde el mismo comienzo de la lógica.

El estudio de la cuantificación se remonta a la época de la Antigua Grecia. En concreto, a los trabajos de Aristóteles sobre silogismos con expresiones cuantificadas. Como menciona Ingo Glöckner en [71], aunque Aristóteles ya hiciera referencia a la cuantificación en sus silogismos, "la lógica aristotélica atañe a la cuantificación universal y existencial, pero carece de las nociones de cuantificador universal y existencial". Es decir, estos cuantificadores no eran tratados como abstracciones lógicas, sino como elementos estructurales de los silogismos. Todo esto dificultó el desarrollo de una lógica con cuantificadores.

Los cuantificadores fueron formalizados a finales del siglo XIX por Gottlob Frege, en su obra de 1879, *Begriffsschrift*, que se puede traducir co-

mo *conceptografía* [64]. En esta obra, Frege presentó una teoría de la cuantificación donde, entre otras cosas, formalizó, a través de una notación bastante enrevesada, los cuantificadores existencial y universal, sentando las bases de un tratamiento más formal de la lógica. Para Frege, los cuantificadores eran un medio para describir las relaciones entre conceptos, aportando la idea de que no solo se podían aplicar a individuos, sino también a funciones y conceptos. Otros autores que contribuyeron a sentar las bases de la teoría de la cuantificación, introduciendo las notaciones actuales para los cuantificadores de primer orden, entre otras cosas, fueron Charles Sanders Peirce en *On the Algebra of Logic: A Contribution to the Philosophy of Notation* [152], Giuseppe Peano en *Formulaire de mathématiques* [151], Bertrand Russell y Alfred North Whitehead en *Principia Mathematica* [200], Gerhard Gentzen en *Untersuchungen über das logische Schließen. I* [67], y Alfred Tarski en *Introduction to Logic and to the Methodology of Deductive Sciences* [188].

Los cuantificadores universal y existencial eran los que se consideraban en los distintos trabajos de la época, y para el estudio de otros cuantificadores, estos se intentaban definir a partir de los cuantificadores de primer orden. De hecho, uno de los aspectos que más ha interesado en la teoría de la cuantificación es el estudio del poder expresivo de los cuantificadores de primer orden, los tradicionales, esto es, si es posible definir el significado de los cuantificadores del lenguaje natural a partir de los cuantificadores existencial y universal de la lógica de predicados. Por ejemplo, frases como "Algunos ganadores de *Roland Garros* no son españoles" pueden reformularse en términos de la cuantificación tradicional como "Existen ganadores de *Roland Garros* que no son españoles".

Sin embargo, esto no se puede hacer en general, y frases como "La mayoría de los pueblos nativos americanos han perdido gran parte de sus territorios históricos" o "Más de la mitad de las exportaciones de petróleo mundiales provienen de Arabia Saudí" contienen expresiones cuantifica-

das que no pueden traducirse en términos de los cuantificadores existencial o universal. El problema con estas expresiones es que no pueden traducirse fácilmente en términos lógicos, como sí ocurre con el universal y el existencial y, por tanto, sus definiciones van a tender a la incertidumbre y la ambigüedad. En los ejemplos que hemos puesto, no es sencillo precisar la cantidad de pueblos nativos a la que se refiere con "la mayoría", ni tampoco está claro la cantidad de exportaciones en la segunda frase. Por tanto, es evidente que, a pesar de la importancia de los cuantificadores de primer orden, su alcance no es suficiente para abarcar todas las formas de cuantificación presentes en el lenguaje natural y en la lógica [116].

A pesar de esta limitación en el poder expresivo de la cuantificación tradicional, algunos cuantificadores como "cada uno", "algún", "no todos" e incluso proposiciones como "Existen al menos k elementos que tienen la propiedad X", para algún cardinal $k \in \mathbb{N}$, sí pueden escribirse a partir de los cuantificadores de primer orden. Este tipo de cuantificadores reciben el nombre de *absolutos*, siendo aquellos que, como hemos visto, expresan una cantidad exacta y se basan en cardinales. Por ejemplo, la frase anterior "Existen al menos k elementos que tienen la propiedad X" puede ser entendida como que la cardinalidad del conjunto de los elementos que tienen la propiedad X es mayor o igual que k. Johan van Benthem demostró que estos cuantificadores se pueden definir en términos de los cuantificadores de primer orden en [195].

Por otro lado, existen otros enunciados como "La mayoría de los kurdos viven en Turquía", cuya verdad depende de la proporción relativa de los kurdos que viven en Turquía con respecto a la población total de kurdos. A estos cuantificadores se les llama *relativos*, y no son definibles en general a partir de los cuantificadores de primer orden, a excepción de algunos casos como "100 por ciento" o "más del 0 por ciento". Por tanto, existen una serie de cuantificadores que no se pueden expresar a partir de la cuantificación de primer orden, a lo que Jon Barwise y Robin Cooper

añaden, en [13], que "no es solo que no sepamos cómo expresarlos en términos de $\forall$ y $\exists$; simplemente no se puede hacer. Por lo tanto, una teoría semántica para el lenguaje natural no puede basarse únicamente en la lógica de predicados". Además, existe una diferencia estructural entre los cuantificadores lógicos que sí son definibles en términos de los de primer orden y los cuantificadores del lenguaje natural, lo que hace que sea necesaria una noción generalizada de los cuantificadores que permitan capturar mejor la expresividad y el significado de los cuantificadores del lenguaje natural. Por ejemplo, en lógica, los cuantificadores suelen ser no restringidos y operan con un único argumento, como en la expresión "Existe un libro". Por el contrario, en el lenguaje natural, los cuantificadores suelen ser restringidos, es decir, limitan el dominio al que se refieren mediante condiciones adicionales y, a menudo, implican dos o más argumentos, como en "La mayoría de los kurdos viven en Turquía", donde nos restringimos al pueblo kurdo [153].

En 1957, el lógico polaco Andrej Mostowski publicó *On a generalization of quantifiers* [136], donde constató las limitaciones de los cuantificadores tradicionales y, como respuesta, presentó una primera extensión de los cuantificadores clásicos, dando comienzo al estudio de la Teoría de Cuantificadores Generalizados. Esta primera generalización corresponde a cuantificadores del tipo $< 1 >$, que tienen un solo argumento y están definidos en un dominio general. Se basan principalmente en los conceptos de cantidad y cardinalidad de conjuntos, y son invariantes bajo isomorfismos, lo que implica que su interpretación no depende de las identidades específicas de los elementos del dominio, sino únicamente de su estructura o de las propiedades abstractas de los conjuntos, como su cardinalidad.

Los cuantificadores de Mostowski sirven para tratar casos como "al menos 5 cosas", "un número par de cosas", "mayor que 7", "un número finito de elementos" o "infinitamente muchos", que son cuantificadores que dependen del cardinal de un conjunto y solo tienen un argumento.

Además, son no restringidos, en el sentido de que las palabras "cosas" o "elementos" hacen referencia a un objeto arbitrario del conjunto, pero no están imponiendo una restricción, lo que sí ocurriría con expresiones como "Al menos 5 barcos" o "un número par de abedules", que no podrían formalizarse con los cuantificadores de Mostowski. El problema reside en el hecho de que estos cuantificadores no tienen el mecanismo necesario para integrar estas restricciones de forma sistemática como lo hace el lenguaje natural.

Aunque esta primera generalización aumentó la expresividad y flexibilidad de los cuantificadores lógicos, seguía siendo limitada para el tratamiento de la cuantificación del lenguaje natural. Como hemos visto, no se tenían en cuenta aspectos como las restricciones, la presencia de varios argumentos, o la no cuantitatividad de ciertos cuantificadores. Con el objetivo de generalizarlos, años más tarde, en 1966, el matemático sueco Per Lindström, en su obra *First Order Predicate Logic with Generalized Quantifiers* [115], introduce una generalización más poderosa de los cuantificadores que hace posible tratar con relaciones binarias o n-arias, es decir, con cuantificaciones que aceptan varios argumentos. Esta extensión corresponde a cuantificadores del tipo $< 1, 1 >$, para el caso de dos argumentos, siendo posible, como hemos dicho, extenderlo a más argumentos. Además, son capaces de vincular más de una variable al mismo tiempo en el cuantificador. Por otro lado, como se menciona en [153], Lindström observó que no era necesario limitar la definición a relaciones entre conjuntos, ya que el concepto podía ampliarse a relaciones entre relaciones, dando lugar a lo que sería la distinción entre cuantificadores monádicos y poliádicos. Ejemplos de cuantificadores que se podían tratar desde la perspectiva de Lindström son "Cada país tiene su bandera" o "Toda ciudad tiene al menos un par de monumentos".

Otros trabajos sobre cuantificadores generalizados fueron presentados por Jerome Keisler en *Logic with the quantifier "there exist uncountably many"*,

en 1969 [108], y por Joseph Sgro, con *Completeness theorems for topological models*, en 1977 [174]. En 1981, Jon Barwise y Robin Cooper publicaron *Generalized Quantifiers and Natural Language* [13], donde presentaron la noción de cuantificador generalizado para una teoría del lenguaje natural, mientras que en *Quantifiers in Language and Logic* [153], Stanley Peters y Dag Westerståhl, realizaron un amplio repaso de la teoría de la cuantificación, incluyendo el desarrollo de los conceptos modernos de la cuantificación, así como de los cuantificadores generalizados.

Aunque los cuantificadores generalizados introducidos hasta el momento supusieron un aumento del poder expresivo, todavía existían limitaciones, puesto que estos cuantificadores trabajan con datos definidos de manera precisa y clara. No obstante, el lenguaje natural es diferente. No es razonable que el valor de verdad de una expresión cuantificada cambie abruptamente al modificarse ligeramente el número de individuos que cumplen una determinada propiedad. Por ejemplo, si consideramos la frase "Mucha gente escucha música" y el número de personas que escuchan música aumenta en una unidad, entonces no podemos esperar que el valor de verdad de la frase pase de verdadero a falso. Por tanto, para poder desarrollar una noción de cuantificadores práctica, es necesario tener en cuenta la ambigüedad del lenguaje natural.

Este fue el motivo por el cual los investigadores en la teoría de la cuantificación consideraron más de dos valores de verdad, a partir de la extensión del conjunto de valores de verdad a un retículo más general, dando lugar a la Teoría de la Cuantificación Difusa. A partir de 1975, Lotfi A. Zadeh introdujo los cuantificadores difusos en una serie de artículos entre los que destaca *PRUF - A meaningful representation language for natural languages* [216], aunque no los trata de manera formal y exclusiva hasta 1983, en el artículo *A computational approach to fuzzy quantifiers in natural languages* [220]. En particular, a partir de la noción de conjunto difuso [218], este tipo de cuantificadores se trata como números difusos. Por ejemplo, el

cuantificador difuso "la mayoría" en la frase "La mayoría de las personas eran respetuosas, pero poco dispuestas a defender lo que él creía" se interpreta como una proporción difusa del conjunto difuso de personas poco dispuestas a defender lo que él creía en el conjunto difuso de personas respetuosas que lo conocían [220].

Asimismo, Zadeh introduce la distinción entre los cuantificadores absolutos y relativos, propone un razonamiento silogístico con cuantificadores difusos, y expone cómo el uso de medidas de cardinalidad generalizadas al marco difuso pueden ser empleadas para implementar cuantificación difusa. Con el objetivo de poder contar con una estructura clara de la cuantificación difusa, útil para su tratamiento, los investigadores Etienne Kerre y Yaxin Liu presentaron una clasificación de las proposiciones cuantificadas difusas en [116]. Distinguieron los siguientes tipos:

- Tipo I: el cuantificador es preciso, es decir, no es difuso, y todos los argumentos son claros, como "Todos los pájaros tienen alas".
- Tipo II: el cuantificador es preciso, pero hay argumentos que son difusos, como "Todos los deportistas de élite entrenan muchas horas".
- Tipo III: el cuantificador es difuso, pero todos los argumentos son claros, como "La mayoría de los pájaros tienen alas".
- Tipo IV: tanto el cuantificador como los argumentos pueden ser difusos, como "La mayoría de los deportistas de élite entrenan muchas horas".

Posteriormente, Zadeh continuó sus trabajos en esta área mejorando el razonamiento difuso con cuantificadores difusos [222], y conectándolos con el tratamiento de la incertidumbre en sistemas expertos [217]. Con el trascurrir de los años, las investigaciones sobre la cuantificación difusa se han centrado principalmente en tres aspectos. El primero de ellos está relacionado con la interpretación de los cuantificadores, es decir, describir

el significado de estos cuantificadores difusos. Como se menciona en [71], "disponer de tales descripciones es un paso previo fundamental para implementar cuantificadores difusos en los ordenadores". Ejemplos de estas investigaciones son los trabajos [221, 220, 158, 159, 209, 211, 116, 25]. En segundo lugar, relacionadas con la manipulación de expresiones con cuantificadores difusos, siendo el objetivo deducir nuevos conocimientos a partir de un conjunto de hechos y reglas que contengan estos cuantificadores difusos. Ejemplos de estas investigaciones son [55, 182, 190, 207, 208, 116]. Por último, las investigaciones de los cuantificadores difusos se centraron en el problema de generar expresiones cuantificadas que describan de manera más precisa una situación dada y su aplicación en sistemas para la extracción de información de bases de datos y para la síntesis y reducción de datos. Entre los ejemplos de estas investigaciones se encuentran [101, 160, 210, 206]

Los últimos avances en cuantificación difusa comprenden los trabajos de Ingo Glöckner que, en 2004, en el artículo [72], extendió la Teoría de Cuantificadores Generalizados incorporando principios de la lógica difusa. Entre otras cosas introdujo un mecanismo de generalización difusa (*fuzzification*, en inglés) de cuantificadores, un método sistemático para generalizar cuantificadores al caso difuso usando, como paso intermedio, cuantificadores semidifusos.

En 2016, Dolores Ruiza, Daniel Sánchez y Miguel Delgado realizaron un estudio en [165] sobre la relación entre la Teoría de Cuantificadores Difusos propuesta por Zadeh y la Teoría de Cuantificadores Generalizados difusos, llegando a la conclusión de que existe una fuerte conexión entre los mecanismos y nociones usadas en ambas teorías. En 2008, Michal Holčapek, estudió los cuantificadores monádicos L-difusos del tipo $< 1^n, 1 >$ en [92], incluyendo la cuantificación difusa en la Teoría de Cuantificadores Generalizados. Más tarde, en 2009, como continuación del trabajo anterior y del de Ingo Glöckner en [70], Antonín Dvořák y Michal Holčapek es-

tudiaron una clase especial de cuantificadores L-difusos de tipo $< 1 >$, en concreto, cuantificadores difusos determinados por medidas difusas, en [59]. Para ello, presentaron nuevos tipos de integrales difusas en las que se basaban las definiciones de los cuantificadores introducidos, a saber, la $\rightarrow$-integral difusa y la $\otimes$-integral difusa, siendo esta última, además, una generalización de la integral de Sugeno [187], y es el concepto en el que se basan los cuantificadores L-difusos de tipo $< 1 >$ determinados por medidas difusas. Estos cuantificadores han sido usados para el estudio de composiciones relacionales difusas basadas en cuantificadores generalizados [197], pero también para sistemas flexibles de respuesta a consultas y bases de datos relacionales difusas [154, 57, 50, 23]. En función de la medida difusa considerada, cuya elección depende de las necesidades del usuario, diferentes cuantificadores generalizados pueden ser considerados. Con el objetivo de reducir la posible presencia de ruido en los datos, debilitaremos el carácter estricto del supremo utilizando cuantificadores generalizados. Nos centraremos en los cuantificadores L-difusos de tipo $< 1 >$ determinados por medidas difusas. Dependiendo de la medida difusa que se utilice, se obtiene un cuantificador general que permite simular los distintos cuantificadores que se pueden usar en el lenguaje natural, distintos del existencial y universal, lo que aporta un nivel más de flexibilidad en la representación lógica del lenguaje natural.

1.1.9. Operadores formales de agregación

En un contexto en el que se encuentran presentes muchos datos y valores, como en cualquier registro o base de datos de una empresa, resulta interesante encontrar un valor que pueda resumir toda la información presente en la distribución. Para ello necesitamos disponer de un método para "convertir" toda esa cantidad de valores en un único dato representativo. Esta idea es la que subyace tras el concepto de *operador formal de agregación*,

o simplemente *agregador formal* u *operador de agregación*. Se trata de operadores matemáticos que sirven para sintetizar información a partir de un conjunto de datos, obteniendo un valor final más significativo del que se habría obtenido al considerar cada una de las piezas de información por separado, y que se utilizan en multitud de campos como en lógica, aprendizaje automático, minería de datos, estadística, inteligencia artificial, sistemas basados en el conocimiento, planificación, controladores de lógica difusa, ingeniería, etc. [24, 168, 180, 213, 145].

Los primeros operadores de agregación aparecen en los trabajos de estadística y probabilidad del matemático suizo Jacob Bernoulli, en su obra publicada de manera póstuma en 1713, *Ars Conjectandi* [17]. Aunque no se nombra específicamente este concepto, sí aparecen algunas nociones que pueden ser consideradas las precursoras de las ideas que darían lugar al desarrollo de los operadores de agregación, como la "Ley de los Grandes Números", que establece que, a medida que el número de repeticiones de un experimento aleatorio aumenta, la media aritmética de los resultados aproxima al valor esperado del experimento, es decir, a la media teórica.

Uno de los campos que contribuyó al avance de la idea de los operadores de agregación fue la Teoría de la Elección Social [9], planteada en medio de la reflexión política iniciada en Francia a finales del siglo XVIII, y cuyo objetivo es el estudio de procesos que permitan a una sociedad elegir entre varias opciones respetando las preferencias individuales de sus miembros. Entre los primeros autores que la abordaron destacan el filósofo balear Ramon Llull [79] y el marqués de Condorcet, matemático francés, que abordó este problema sobre la agregación de las preferencias individuales, introduciendo, entre otras cosas, la conocida como *paradoja de Condorcet* en 1785, en una época de crisis del antiguo régimen, con una sociedad marcada por profundos cambios políticos, económicos y sociales, que culminaría en la revolución del 5 de mayo de 1789 [39]. La paradoja de Condorcet mostró que existen situaciones donde las preferencias individuales no se pueden

fusionar de manera tan coherente en una única preferencia grupal, dando lugar a más estudios sobre la agregación de preferencias y, por ende, en el desarrollo de los operadores formales de agregación.

El trabajo iniciado por el marqués de Condorcet alcanzó su plenitud a partir de 1951 con el estudio moderno de la Teoría de la Elección Social realizado por el matemático y economista estadounidese Kenneth Arrow en *Social Choice and Individual Values* [9]. En esta obra, Arrow se plantea si existe algún método de agregación óptimo que permita dar lugar a un resultado colectivo satisfactorio para todas las preferencias individuales, llegando a la conclusión de que en un sistema de elección entre tres alternativas, con supuestos aceptables, lógicos y de sentido común, no existe ninguna agregación que conduzca a un resultado de bienestar social, es decir, que satisfaga las tres alternativas, a partir de las diferentes preferencias individuales, sin evitar que un individuo tenga que imponer su criterio; reflejado todo ello en el conocido Teorema de Imposibilidad de Arrow [9].

Otra de las ramas que ha contribuido al desarrollo de los operadores de agregación es la *Teoría de la Toma de Decisiones Multicriterio*, influenciada también por la Teoría de la Elección Social a través de conceptos clave como la agregación de preferencias en su búsqueda de optimizar la toma de decisiones en situaciones electorales. Tras la Segunda Guerra Mundial, los avances tecnológicos, junto con el auge de la investigación de operaciones como una disciplina sólida para resolver problemas de decisión, dieron lugar a un nuevo marco para abordar desafíos del mundo real a través de métodos más integrados, flexibles y prácticos [54]. Todo esto impulsó el desarrollo y la consolidación, entre otras cosas, de la Teoría de la Toma de Decisiones Multicriterio, que "constituye un campo avanzado de investigación que se dedica al desarrollo e implementación de herramientas y metodologías de apoyo a la toma de decisiones para tratar problemas de decisión complejos que involucran múltiples criterios, metas u objetivos

de naturaleza conflictiva" [104].

Normalmente, los principales factores que se deben tener en cuenta en un proceso de toma de decisión, y que deben estar especificados con claridad, son un conjunto de alternativas, un conjunto de restricciones o criterios para elegir entre las distintas alternativas y un conjunto de consecuencias asociadas a la elección de cada una de estas alternativas. No obstante, muchas de las situaciones de toma de decisión en la realidad suceden en ambientes donde estas componentes son imprecisas, como la elección de un piso que esté cerca del centro de Jerez de la Frontera, o que el coste de realizar un determinado trabajo no sea excesivamente elevado. Por esta razón, en 1970, R. E. Bellman y L. A. Zadeh introdujeron y estudiaron el concepto de toma de decisiones difusas en *Decision-making in a fuzzy environment* [15]. El problema de tomar una decisión en un contexto difuso consiste en agregar los diferentes valores de verdad que expresan la relación existente entre cada criterio o restricción y las alternativas.

Desde entonces, numerosos artículos se han publicado sobre los operadores de agregación, en los que se han propuesto diferentes formas de realizar las agregaciones. Destacan los trabajos *Theory of capacities*, de Gustave Choquet [34], y *Theory of fuzzy integrals and applications*, de Michio Sugeno [187], en los que se introducen métodos de agregación basados en integrales, que posteriormente serán ampliados y aplicados en libros como *Fuzzy integral in multicriteria decision making* [74] o *Aggregation operators: new trends and applications* [24]. Además de estas técnicas de agregación, también se pueden considerar métodos más sencillos como los operadores máximo y mínimo, aunque en este caso se pierde información puesto que si, por ejemplo, tenemos que agregar los valores 0.1 y 0.9 con el mínimo, nos daría el valor 0.1, perdiéndose toda la información que representa el 0.9. Esta es la razón por la que se introdujeron en el contexto de los conectores lógicos difusos las normas triangulares, o t-normas, y las t-conormas, que fueron planteadas inicialmente en 1942 por Karl Menger

con respecto al estudio de espacios métricos estadísticos [132], y estudiadas, posteriormente, por Schweizer y Sklar en [171]. Las t-normas y las t-conormas han demostrado ser generalizaciones apropiadas de los conectores lógicos clásicos, la conjunción y la disyunción, respectivamente, al ambiente difuso [56].

A pesar de la gran cantidad de trabajos existentes sobre la agregación, los investigadores Tomasa Calvo, Gaspar Mayor y Radko Mesiar señalan en [24], que "no hay una notación unificada en el campo de la agregación. La mayoría de los resultados están diseminados en diferentes artículos, donde generalmente se discute una clase específica de operadores de agregación". Por esta razón no se puede dar una única definición de operador de agregación. Seguiremos, pues, la dada por George Klir y Bo Yuan en [109], que definen formalmente un operador de agregación sobre conjuntos difusos como una función $h\colon [0,1]^n \longrightarrow [0,1]$, donde $n \geq 2$ es el número de valores a agregar. Esta función agrega los grados de pertenencia de los conjuntos difusos, de tal forma que el conjunto difuso agregado A que se obtiene verifica que $A(x) = h\left(A_1(x), A_2(x), \ldots, A_n(x)\right)$, donde $A_1, A_2, \ldots, A_n$ son conjuntos difusos definidos en X, y $x \in X$. Además, estos operadores cumplen una serie de propiedades.

Una de las propiedades que deben satisfacer los operadores de agregación son las condiciones de frontera, preservando las acotaciones del conjunto de partida. Esta propiedad se exige para que la agregación respete la minimalidad o maximalidad de los datos de entrada. Por tanto, el operador de agregación debe satisfacer que $h(0,\ldots,0) = 0$ y $h(1,\ldots,1) = 1$. Además, para preservar la relación de orden entre los valores de entrada, se exige que el operador de agregación sea monótono, es decir, si tenemos dos tuplas de valores de entrada $(x_1,\ldots,x_n), (y_1,\ldots,y_n) \in [0,1]^n$, tal que $x_i \leq y_i$, para todo $i \in \{1,\ldots,n\}$, entonces $h(x_1,\ldots,x_n) \leq h(y_1,\ldots,y_n)$. Existen también otras aproximaciones a la monotonía de los operadores de agregación, que son menos restrictivas.

Una de las propiedades más interesantes que debe cumplir un operador de agregación es la idempotencia, es decir, que $h(x, \ldots, x) = x$. En el contexto de la Teoría de la Toma de Decisiones Multicriterio, esto implica que si todos los criterios se satisfacen con un grado x, entonces la decisión global también debe satisfacerse con un grado x. Estos operadores de agregación idempotentes también se llaman operadores ponderados o promedio. Si aceptamos la idempotencia como una propiedad crucial, entonces cualquier agregador siempre va a estar acotado entre los agregadores mínimo y máximo, relacionados con las condiciones de frontera.

También es deseable que nuestro operador de agregación sea continuo, especialmente en el ámbito de la ingeniería, para que pequeños errores o variaciones en los valores de entrada, no produzcan grandes errores en el valor final. Finalmente, si todos los conjuntos difusos que se están agregando son igualmente importantes, se podría pedir que este agregador fuese simétrico, aunque lo usual en el contexto de la toma de decisiones multicriterio es asignar a cada uno de los valores de entrada un peso específico al agregarlo en función de algún juicio, por lo que la simetría no sería un requisito fundamental.

Relacionado con este último punto, un factor fundamental que se debe tener en cuenta en la elección de los operadores de agregación, para que puedan dar lugar a una decisión conveniente, es la relación que existen entre los criterios. Por un lado, tenemos el caso extremo en el que queremos que todos los criterios se satisfagan, en cuyo caso, para agregar todas las preferencias se utilizaría el operador "y" (*and*), que cuantitativamente se interpreta utilizando t-normas. En cambio, por otro lado, tenemos el otro caso extremo en el que buscamos la satisfacción de cualquiera de los criterios. Para agregar estos criterios usamos el operador "o" (*or*) que, desde un enfoque cuantitativo, se interpreta empleando t-conormas. No obstante, se trata de situaciones singulares y existen casos intermedios más interesantes en multitud de aplicaciones.

Para flexibilizar este proceso de agregación, en 1987, el americano Ronald R. Yager, especializado en áreas como lógica difusa, funciones de agregación y teoría de toma de decisiones, introdujo una nueva familia de operadores llamados operadores de promedio ponderado ordenado, simplificado operadores OWA, del inglés *Ordered Weighted Averaging operators*, en *On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking* [209]. Al contrario de lo que sucede con el promedio ponderado clásico en el que los coeficientes de los pesos se asocian directamente con un atributo, en los operadores OWA estos coeficientes se asignan según una posición ordenada, relacionándolo con la idea de que estos agregadores combinan los criterios bajo la influencia de los cuantificadores.

Los operadores OWA verifican propiedades interesantes como las condiciones de frontera, la monotonía, la simetría, la continuidad o la idempotencia. Además, mientras que los cuantificadores L-difusos de tipo $< 1 >$ determinados por medidas difusas (Sección 1.1.8) son un caso particular de integral de Sugeno; Calvo, Mayor y Mesiar demuestran en [24] que cada operador OWA puede ser representado en términos de la integral de Choquet con respecto a un sistema apropiado de medidas difusas simétricas.

Todas estas propiedades convierten a los operadores OWA en herramientas flexibles y eficientes para la agregación de información en diversas situaciones. Entre estos contextos destacan su uso en el modelado de redes neuronales [202], sistemas de base de datos [212], sistemas expertos [143], el desarrollo de un sistema de programación neuronal de lógica difusa [60], sistemas difusos y de control [205, 214], la toma de decisiones [209, 2, 32, 103], sistemas de recomendación [85, 178, 223], clasificación [121], minería de datos [192, 203, 204], consulta de bases de datos [102], predicción [204], razonamiento aproximado [31], etc. Posteriormente, Vicenç Torra propuso el concepto de operador WOWA, que es una

generalización tanto de los operadores OWA como de la media ponderada, permitiendo la agregación de datos atendiendo a dos vectores de pesos, uno que se corresponde con el de la media ponderada, y otro que es el del operador OWA [193].

1.2. Hipótesis y objetivos

A continuación, se expondrá la justificación principal del trabajo, motivada por la problemática de la definición del $T_{\mathbb{P}}$ a partir del supremo. El operador de consecuencias inmmediatas aplicado sobre una variable A agrega la información dada por las reglas cuyas cabezas están relacionadas con A, ya sea porque tengan esa misma cabeza, como en lógica proposicional, o porque las cabezas de las reglas unifican con A, como en lógica de predicados. Esta agregación se realiza a partir del operador supremo, es decir, la menor de las cotas superiores de los valores aportados por las reglas. El valor que se obtiene en la última iteración del $T_{\mathbb{P}}$ es el que se asocia como valor de verdad a A. No obstante, la naturaleza estricta del supremo, que no tiene en cuenta la mayoría de los valores agregados, puede ocasionar que la posible presencia de ruido en los datos o una calidad de datos deficiente, cause alteraciones significativas en los resultados finales, pudiendo generar desde pérdidas de ingresos hasta funcionamientos incorrectos de sistemas inteligentes.

Supongamos que tenemos varias reglas lógicas que están modelizando un sistema inteligente de aprobación de transacciones para un banco y que los valores de verdad aportados por cada una de las reglas son 0.30, 0.29, 0.31, 0.28, pero se ha producido un error, ya sea por falta de precisión o porque no están actualizados algunos datos de la base de datos y la última regla da un valor de 0.71. A partir de la definición del operador de consecuencias inmediatas como el supremo del conjunto de valores anteriores, nos daría como valor 0.71, a pesar de que la mayor parte de los valores

obtenidos desaconsejaban realizar la transacción. Se puede ver, por tanto, la gran cantidad de información que no es tenida en cuenta y que puede conducir a tomar decisiones erróneas.

Teniendo en cuenta esta problemática, el propósito del presente trabajo es reducir el carácter estricto del operador de consecuencias inmediatas proponiendo generalizaciones del mismo que sean capaces de trabajar en bases de datos con incertidumbre y con la posible presencia de ruido en los datos. Estas extensiones de dicho operador se basarán en la utilización de cuantificadores generalizados y operadores de agregación promedio ponderados.

Además de estudiar una semántica de punto fijo asociada a la nueva definición del operador de consecuencias inmediatas, también se estudiará una semántica operacional, a partir de la cual se podrán diseñar algoritmos que permitan deducir consecuencias de los sistemas modelados.

Otro de los objetivos planteados es el estudio de la versatilidad del uso de distintas implicaciones en la construcción de las reglas lógicas del programa para su aplicación en la asignación de distintos grados de preferencia en sistemas de ayuda a la toma de decisiones. Esto último se ha presentado en el capítulo titulado *Preference in Multi-adjoint Logic Programming Based on Ordered Adjoint Pairs* [127], que forma parte del volumen publicado con las versiones definitivas de los artículos seleccionados del congreso de matemáticas *14th European Symposium on Computational Intelligence and Mathematics* (ESCIM 2022).

Capítulo 2

Discusión conjunta de los resultados

En este capítulo presentamos las principales contribuciones de la investigación realizada en la presente tesis doctoral. Estos resultados se han publicado en los artículos [129, 122, 131, 130], o se han enviado para su publicación en una revista de impacto [128].

2.1. Operador de consecuencias inmediatas con cuantificadores generalizados

Como hemos comentado en la Sección 1.1 del capítulo anterior, las bases de datos pueden presentar una mala calidad de datos, ya sea por la falta de información, que se manifiesta a través de la existencia de datos incompletos, o por la presencia de "ruido" en los datos, es decir, de información que es errónea. Tal como se observó en la Sección 1.2, esta mala calidad de los datos puede provocar que el supremo en la definición del operador de consecuencias inmediatas $T_{\mathbb{P}}$ proporcione valores que no sean

fiables, provocando que la decisión o el valor final obtenido sea erróneo o, al menos, alejado de la realidad. En consecuencia, este artículo tiene como objetivo presentar una definición alternativa del operador de consecuencias inmediatas utilizando cuantificadores generalizados para flexibilizar el carácter estricto del supremo.

Para ser preciso, en el artículo *Immediate consequences operator on generalized quantifiers* [129], publicado en 2022 en la revista *Fuzzy Sets and Systems*, cuyo factor de impacto lo sitúa en el primer cuartil (Q1) de su categoría del *Journal Citation Reports (JCR)*, se propone y estudia una nueva definición del operador de consecuencias inmediatas, en el contexto de la programación lógica proposicional multiadjunta, basada en cuantificadores generalizados determinados por una medida difusa μ, de los cuales se ha tratado previamente en la Sección 1.1.8. En función de la medida difusa considerada, estos cuantificadores están comprendidos entre los dos casos extremos correspondientes a los cuantificadores de primer orden, es decir, el existencial y el universal. Por tanto, una gran variedad de diferentes operadores de consecuencias inmediatas pueden obtenerse considerando diversas medidas difusas en función de las condiciones del entorno, las necesidades del usuario, las directrices de los técnicos y de los expertos, etc. Lo relevante es que se ha definido un nuevo operador $T_{\mathbb{P}}$, que es fundamental para el estudio de la semántica de los programas lógicos, y es mucho más flexible que el operador original, lo cual añade mayor expresividad al lenguaje, permitiendo adaptarse a situaciones más complejas.

Siguiendo lo establecido en la Sección 1.2, si consideramos la definición usual del $T_{\mathbb{P}}$ basada en el supremo, junto con el programa lógico $\mathbb{P}$ dado en [129, Ejemplo 2], y calculamos el valor del operador de consecuencias inmediatas para la interpretación I dada en [129, Ejemplo 5] y el símbolo proposicional `covid`, llegamos al resultado final $\sup\{0.81, 0.49, 0.5\}$. Dos de estos valores están más próximos entre sí, mientras que el tercero está, relativamente hablando, mucho más alejado dentro del conjunto de valo-

res de verdad $[0, 1]$. Aunque todos estos datos pudieran ser correctos, no dejaría de sorprender que, en la toma de una decisión, uno de los valores obtenidos, para el mismo símbolo proposicional, esté tan alejado de los demás. A pesar de que se trata de un ejemplo limitado y prácticamente desprovisto de relevancia desde una perspectiva sintáctica (estructural), resulta adecuado para ilustrar, de manera didáctica, la enorme sensibilidad del operador de consecuencias inmediatas frente a errores presentes en las bases de datos que modelan las reglas lógicas o a fallos en el proceso de observación, a partir del cual se definen los hechos del programa. Lo cierto es que resulta necesario modificar esta definición para que podamos paliar de alguna manera la presencia de datos disonantes en nuestras bases de datos. Para ello, en este artículo, como hemos mencionado, aportamos una nueva definición basada en los cuantificadores generalizados determinados por medidas difusas.

El primer paso para afrontar esta tarea consiste en adaptar la definición clásica del operador de consecuencias inmediatas utilizando los cuantificadores generalizados. En concreto, nos damos cuenta de que definiendo el cuantificador existencial $Q^{\exists}$ sobre la aplicación $C_I^A \in \mathcal{F}([0, 1])$ dada en [129, Definición 12], obtenemos la definición tradicional del $T_{\mathbb{P}}$ en el marco multiadjunto [124]. Además, este resultado permite asegurar que la nueva definición que se va a proponer es una generalización de la original, puesto que esta última se obtiene como un caso particular. El operador C_I^A, definido sobre el conjunto de reglas del programa, aporta el valor de la interpretación I sobre una regla $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$, es decir, $\vartheta \dot{\&}_i \hat{I}(\mathcal{B})$, que se obtiene a partir del teorema de extensión homomórfica y la propiedad de adjunción [124]. Para una misma interpretación I, en un programa lógico $\mathbb{P}$ tenemos tantas aplicaciones C_I^A como símbolos proposicionales cabezas de una regla en $\mathbb{P}$.

Específicamente, la nueva definición del operador de consecuencias inmediatas se da a continuación. El detalle de los operadores y símbolos que

se utilizan se puede consultar directamente en el artículo [129] o en el Capítulo 3.

Definición 2.1. *Sea $\mathbb{P}$ un programa lógico multiadjunto, $\mu\colon \mathcal{P}(\mathbb{P}) \longrightarrow [0,1]$ una medida difusa invariante con respecto a la cardinalidad y $\otimes$ un operador creciente en el primer argumento y que verifica las siguientes condiciones de frontera: $x \otimes 1 = x$, $x \otimes 0 = 0 \otimes x = 0$, para todo $x \in [0,1]$. Definimos el* operador de consecuencias inmediatas cuantificado $T_{\mathbb{P}}^{Q_\mu} : \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$*, obtenido del cuantificador generalizado Q_μ, como*

$$T_{\mathbb{P}}^{Q_\mu}(I)(A) = Q_\mu\left(C_I^A\right) = \bigvee_{D \in \mathcal{P}(\mathbb{P})} \left(\left(\bigwedge_{\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in D} C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle\right) \right) \otimes \mu(D) \right)$$

para todo $I \in \mathcal{I}_{\mathfrak{L}}$ y $A \in \Pi$, donde la aplicación $C_I^A\colon \mathbb{P} \longrightarrow [0,1]$ se define, para cada $\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}$, como

$$C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle\right) = \begin{cases} \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) & \text{si } C = A \\ 0 & \text{en caso contrario} \end{cases}$$

Como vemos, estamos definiendo el $T_{\mathbb{P}}$ cuantificado como el cuantificador generalizado Q_μ sobre la aplicación C_I^A. El operador $\otimes$ que introducimos en la definición verifica unas condiciones de frontera con el fin de que esta definición sea generalización de la versión clásica, es decir, para que los cuantificadores existencial y universal se obtengan como casos particulares del cuantificador generalizado.

Este nuevo operador satisface una propiedad fundamental que es la monotonía [129, Corolario 1], preservando este rasgo del caso clásico, que nos dice que si las interpretaciones de las reglas son mayores, entonces el valor obtenido como conclusión del operador de consecuencias inmediatas también será mayor. No obstante, cabe señalar que esta nueva definición no es particularmente manejable, porque implica realizar el supremo de un ínfimo sobre todos los conjuntos de $\mathcal{P}(\mathbb{P})$. Martin Štěpnička y Michal

Holčapek utilizan la propiedad de invarianza con respecto a la cardinalidad de las medidas difusas para introducir en [197] una caracterización de los cuantificadores generalizados a través del supremo de un conjunto de valores ordenados. Esta caracterización será utilizada para simplificar los cálculos asociados al nuevo operador de consecuencias inmediatas.

Con el objetivo de estudiar la semántica del programa a partir de esta nueva definición, comenzamos analizando un caso del $T_{\mathbb{P}}$ cuantificado basado en una medida difusa particular. Una posible solución inicial al problema de la presencia de valores disruptivos en las bases de datos podría consistir en su eliminación o en la aplicación de una penalización significativa, de manera que, al calcular el supremo, el operador de consecuencias inmediatas no tenga en cuenta dichos valores. En el ejemplo particular que se está analizando, consideramos la medida difusa μ^1, definida en el intervalo unidad, y que, en lugar de tener en cuenta el valor más alto, como hace el supremo en un subconjunto finito del intervalo unidad, permita elegir el segundo valor más alto del conjunto. Fijémonos que si estuviéramos trabajando en un retículo más general L, entonces sería más complicado realizar todo este procedimiento considerando que no podemos asegurar que dos elementos cualesquiera de L sean comparables.

Para este caso específico, se estudia la semántica del programa y, para ello, necesitamos una nueva definición de modelo. Originalmente, un modelo es una interpretación que satisface todas las reglas del programa. No obstante, teniendo en cuenta que estamos "eliminando" el valor más alto, una de las reglas puede no ser satisfecha por la interpretación. Por esta razón se define el concepto de μ^1-*modelo débil*, dependiente de la medida difusa μ^1, como una interpretación que satisface todas las reglas del programa, excepto posiblemente una de ellas. Un resultado elemental para el estudio de la semántica de punto fijo es la siguiente caracterización de los μ^1-modelos débiles como puntos posfijos del operador de consecuencias inmediatas cuantificado.

Teorema 2.1. *Sea $\mathbb{P}$ un programa lógico multiadjunto (finito) e I una interpretación de $\mathbb{P}$. Entonces, I es un μ^1-modelo débil de $\mathbb{P}$ si y solo si $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$.*

Demostración. Sea $\{u_1, u_2, \ldots, u_{n-1}, u_n\}$ el conjunto de reglas de $\mathbb{P}$ con cabeza el símbolo proposicional $A \in \Pi$, e I un μ^1-modelo débil de $\mathbb{P}$. Podemos considerar una permutación π en este conjunto de tal forma que la siguiente cadena de desigualdades se satisfaga

$$C_I^A\left(u_{\pi(n)}\right) \leq C_I^A\left(u_{\pi(n-1)}\right) \leq \ldots \leq C_I^A\left(u_{\pi(2)}\right) \leq C_I^A\left(u_{\pi(1)}\right) \tag{2.1}$$

Vamos a demostrar que $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$.

Por un lado, como I es un μ^1-modelo débil, entonces todas las reglas del programa, excepto posiblemente una de ellas, son satisfechas por I. Supongamos que I no satisface la regla $u_{\pi(i)} = \left\langle A \leftarrow_{\pi(i)} \mathcal{B}_{\pi(i)}, \vartheta_{\pi(i)} \right\rangle$, para algún $i \in \{2, \ldots, n\}$. Entonces, se cumple que $\vartheta_{\pi(i)} \not\leq \hat{I}\left(A \leftarrow_{\pi(i)} \mathcal{B}_{\pi(i)}\right)$ o, equivalentemente, usando la propiedad de adjunción y la definición de $\hat{I}$, $\vartheta_{\pi(i)} \dot{\&}_{\pi(i)} \hat{I}\left(\mathcal{B}_{\pi(i)}\right) \not\leq I(A)$, siendo igual a $C_I^A\left(u_{\pi(i)}\right) \not\leq I(A)$.

Por otro lado, la regla $u_{\pi(i-1)} = \left\langle A \leftarrow_{\pi(i-1)} \mathcal{B}_{\pi(i-1)}, \vartheta_{\pi(i-1)} \right\rangle$ debe ser satisfecha por I. Por lo tanto, debe verificarse que $\vartheta_{\pi(i-1)} \leq \hat{I}\left(A \leftarrow_{\pi(i-1)} \mathcal{B}_{\pi(i-1)}\right)$; expresión que es equivalente a $\vartheta_{\pi(i-1)} \dot{\&}_{\pi(i-1)} \hat{I}\left(\mathcal{B}_{\pi(i-1)}\right) \leq I(A)$, que es igual a $C_I^A\left(u_{\pi(i-1)}\right) \leq I(A)$.

Por el orden dado en la Expresión (2.1), obtenemos que $C_I^A\left(u_{\pi(i)}\right) \leq C_I^A\left(u_{\pi(i-1)}\right) \leq I(A)$, llegando a una contradicción con $C_I^A\left(u_{\pi(i)}\right) \not\leq I(A)$. En consecuencia, concluimos que la única regla que puede no ser satisfecha por la interpretación I es $u_{\pi(1)}$. En particular, $u_{\pi(2)} = \left\langle A \leftarrow_{\pi(2)} \mathcal{B}_{\pi(2)}, \vartheta_{\pi(2)} \right\rangle$ es satisfecha por I, es decir, llegamos a la siguiente cadena de desigualdades equivalentes.

$$\begin{aligned} \vartheta_{\pi(2)} &\leq \hat{I}\left(A \leftarrow_{\pi(2)} \mathcal{B}_{\pi(2)}\right) \\ \vartheta_{\pi(2)} \dot{\&}_{\pi(2)} \hat{I}\left(\mathcal{B}_{\pi(2)}\right) &\leq I(A) \\ C_I^A\left(u_{\pi(2)}\right) &\leq I(A) \end{aligned}$$

Teniendo en cuenta el orden dado en la Expresión (2.1) y la elección de la medida difusa μ^1, se demuestra que

$$\begin{aligned} T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) &= Q_{\mu^1}\left(C_I^A\right) \\ &= \bigvee_{i=1}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1,\ldots,u_i\}\right) \\ &= \bigvee_{i=2}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1,\ldots,u_i\}\right) \\ &= C_I^A\left(u_{\pi(2)}\right) \leq I(A) \end{aligned}$$

Hemos probado que $T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) \leq I(A)$, para todo $A \in \Pi$, es decir, $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$.

Ahora, procederemos a probar la implicación recíproca, esto es, si I es una interpretación que satisface $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$, tenemos que demostrar que I es un μ^1-modelo débil. Como $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$, entonces, $C_I^A\left(u_{\pi(2)}\right) = T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) \leq I(A)$. Por el orden dado por π, se cumple que $C_I^A\left(u_{\pi(i)}\right) \leq I(A)$, o equivalentemente, $\vartheta_{\pi(i)} \leq \hat{I}\left(A \leftarrow_{\pi(i)} \mathcal{B}_{\pi(i)}\right)$, para todo $i \in \{2,\ldots,n\}$. Por lo tanto, todas las reglas, excepto posiblemente una de ellas, que es la asociada a $\pi(1)$, son satisfechas por I y podemos concluir que I es un μ^1-modelo débil de $\mathbb{P}$. □

La caracterización de los μ^1-modelos débiles de $\mathbb{P}$ como puntos posfijos del operador $T_{\mathbb{P}}^{Q_{\mu^1}}$, junto con la monotonía de este operador, nos permite demostrar en [Proposición 5][129] que el operador de consecuencias inmediatas cuantificado posee un menor punto fijo. Por lo tanto, se alcanza un punto de estabilidad, que se corresponde con el ínfimo del conjunto formado por todos los μ^1-modelos débiles del programa, que además sabemos que existe porque $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$ forma un retículo completo. Observemos que estamos relacionando el significado del programa, expresado a través del ínfimo de las interpretaciones (concepto semántico), con la noción de punto fijo del operador de consecuencias inmediatas (concepto sintáctico).

Esta proposición se complementa con el siguiente resultado, [129, Teorema 6], en el que establecemos una forma de calcular este punto fijo. En concreto, se demuestra que el menor punto fijo del operador de consecuencias inmediatas cuantificado se puede obtener iterando este operador un número ordinal de veces α, empezando por la menor interpretación $\triangle$. En consecuencia, este procedimiento permite determinar las consecuencias más significativas de un programa lógico.

Aunque esta generalización del $T_{\mathbb{P}}$ ayuda a descartar la posible presencia de ruido en los datos, presenta una desventaja relacionada con la elección de la medida difusa en aquellos casos donde haya símbolos proposicionales que son cabeza de una única regla. Imaginemos, por ejemplo, que tenemos un símbolo proposicional A que es cabeza de varias reglas, mientras que tenemos otro B que es cabeza únicamente de una regla. Además, para determinar la semántica de nuestro programa lógico, nos valdremos del operador de consecuencias inmediatas cuantificado con la medida difusa μ^1. En el primer caso, este operador nos daría el segundo valor más alto de aquellas reglas con cabeza A, mientras que en el segundo se presentan inconvenientes al no existir un segundo valor más alto. En este último caso, el valor del $T_{\mathbb{P}}^{Q_{\mu^1}}$ para el símbolo B sería 0, eliminando toda la información que aportaba esa regla.

Una primera idea para solucionar este contratiempo es modificar la medida difusa. En lugar de "eliminar" el valor más alto, lo que se puede hacer es considerar una medida difusa particular para cada símbolo proposicional cabeza de una regla, es decir, consideramos una familia de medidas difusas

$$M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi, \mathbb{P}_A \neq \varnothing\}$$

donde a cada símbolo proposicional cabeza de una regla en el programa se le asigna la medida difusa más apropiada. De esta forma, obtenemos una generalización del $T_{\mathbb{P}}$ cuantificado ([129, Teorema 7]), que recibe el nombre

de M-operador de consecuencias inmediatas cuantificado, $T_{\mathbb{P}}^{M}$, que además es monótono, como se demuestra en [129, Proposición 6].

Con respecto a la definición de la semántica, dada la complejidad de proveer de una descripción semántica a un cuantificador generalizado arbitrario, es difícil dar una definición de modelo basada en una familia de medidas difusas. Como el objetivo es que la definición de los nuevos modelos cumpla la caracterización de los puntos posfijos del $T_{\mathbb{P}}^{M}$, definimos los M-modelos, dependientes de la familia de medidas difusas considerada, como los puntos posfijos del operador de consecuencias inmediatas cuantificado. Teniendo en cuenta esto, como extensión de los resultados anteriormente probados para el caso particular de la medida difusa μ^1, se demuestra que el menor M-modelo del programa se corresponde con el menor punto fijo del $T_{\mathbb{P}}^{M}$, que además puede ser obtenido computacionalmente iterando el operador $T_{\mathbb{P}}^{M}$ un número ordinal de veces α, empezando por la menor interpretación $\triangle$.

Si bien la definición del operador de consecuencias inmediatas ha sido modificada, no podemos obviar la definición original basada en el supremo. Por tanto, resulta aconsejable que el operador derive valores cercanos al supremo y esto ocurre si, al definir el $T_{\mathbb{P}}^{Q_\mu}$, consideramos medidas difusas μ cuyos valores estén próximos a los de la existencial $\mu^{\exists}$. Para ello, introducimos un indicador del grado de cercanía de las medidas difusas con respecto a la medida existencial, basado en una adaptación, al marco de trabajo en el que nos encontramos, del *índice de optimismo* (*orness*, en los textos en inglés), definido por Ronald Yager en [209], en el contexto de los operadores ponderados de agregación (OWA). En concreto, introducimos el concepto de índice de optimismo asociado a una medida difusa μ, y su generalización, asociada a una familia de medidas difusas M. Cuanto más cercano sea el valor del índice de optimismo a 1, más cerca del supremo está el valor de $T_{\mathbb{P}}^{Q_\mu}$. La idea de esta definición, especialmente del factor multiplicativo, es que el índice de optimismo de la medida existen-

cial sea 1.

Por lo tanto, la medida difusa recomendada al usuario sería aquella que, además de reducir la posible presencia de ruido en los datos específicos a tratar, presente un índice de optimismo tan cercano como sea posible a 1. De esta forma, la selección de una medida difusa adecuada hace del operador de consecuencias inmediatas cuantificado un elemento esencial para disminuir el ruido presente en las bases de datos.

2.2. Alcance computacional de consecuencias cuantificadas a partir de datos iniciales imperfectos

El artículo *Immediate consequences operator on generalized quantifiers* [129] ha sido fundamental para flexibilizar el operador de consecuencias inmediatas en su concepción original en el entorno multiadjunto. Esta generalización posee propiedades destacadas como la monotonía, pero hay otra propiedad que resulta igualmente relevante, esto es, la continuidad. El hecho es que el nuevo operador de consecuencias inmediatas cuantificado puede no ser continuo, necesitándose entonces más de una cantidad numerable de iteraciones para alcanzar el menor punto fijo y, por tanto, el menor modelo del programa. Por consiguiente, la continuidad es una de las propiedades más deseadas para la nueva definición del operador. El presente artículo, *Computational reaching of quantified consequences from imperfect initial data* [122], publicado en 2024 en la revista *Mathematical Methods in the Applied Sciences*, con un factor de impacto que lo ubica en el primer cuartil (Q1) de su categoría del *JCR*, aporta condiciones suficientes y necesarias para asegurar la continuidad de dicho operador.

La condición suficiente se establece en el siguiente resultado, siendo

una de las principales contribuciones del trabajo, que complementa el iniciado en [129]. Los detalles del teorema se pueden consultar en el artículo [122] o en el Capítulo 4.

Teorema 2.2. *Si todos los operadores del cuerpo de las reglas del programa $\mathbb{P}$ son continuos Scott, los conjuntores adjuntos son continuos Scott en el segundo argumento y el operador $\otimes$ es continuo Scott en el primer argumento, entonces $T_{\mathbb{P}}^{M}$ es continuo Scott.*

Demostración. Sea $X \subseteq \mathcal{I}_{\mathfrak{L}}$ un subconjunto dirigido de interpretaciones, $A \in F_{\Omega}$ un símbolo proposicional y $\langle A \leftarrow_i \mathcal{B}_i, \vartheta_i \rangle \in \mathbb{P}$, para todo $i \in \{1, \ldots, n\}$, el subconjunto de reglas con cabeza A. Tenemos que probar que

$$T_{\mathbb{P}}^{\mathrm{M}}\left(\sup X\right)(A) = \sup\left\{T_{\mathbb{P}}^{\mathrm{M}}(J)(A) \mid J \in X\right\} \tag{2.2}$$

Llamando $S = \sup X$, se obtiene la siguiente cadena de igualdades.

$$\begin{aligned}
T_{\mathbb{P}}^{\mathrm{M}}\left(\sup X\right)(A) &= T_{\mathbb{P}}^{Q_{\mu_A}}\left(\sup X\right)(A) \\
&= Q_{\mu_A}\left(C_S^A\right) \\
&= \bigvee_{i=1}^{n}\left(\vartheta_i \dot{\&}_i \widehat{S}(\mathcal{B}_i)\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \\
&\overset{(1)}{=} \bigvee_{i=1}^{n}\left(\vartheta_i \dot{\&}_i \sup\left\{\widehat{J}(\mathcal{B}_i) \mid J \in X\right\}\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \\
&\overset{(2)}{=} \bigvee_{i=1}^{n} \sup\left\{\vartheta_i \dot{\&}_i \widehat{J}(\mathcal{B}_i) \mid J \in X\right\} \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \\
&\overset{(3)}{=} \bigvee_{i=1}^{n} \sup\left\{\left(\vartheta_i \dot{\&}_i \widehat{J}(\mathcal{B}_i)\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \mid J \in X\right\} \\
&= \sup\left\{\bigvee_{i=1}^{n}\left(\vartheta_i \dot{\&}_i \widehat{J}(\mathcal{B}_i)\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \mid J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{Q_{\mu_A}}(J)(A) \mid J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{\mathrm{M}}(J)(A) \mid J \in X\right\}
\end{aligned}$$

donde la igualdad (1) se satisface por [122, Lema 1], mientras que las igualdades (2) y (3) son consecuencia de la continuidad del operador $\&_i$ en el segundo argumento y del operador $\otimes$ en el primero. Por lo tanto, podemos concluir que $T_{\mathbb{P}}^{\mathrm{M}}$ es continuo Scott. □

Una condición necesaria la obtenemos en [122, Teorema 3], en la que se establece que, para que el operador de consecuencias inmediatas cuantificado sea continuo, es indispensable exigir la continuidad de los conjuntores adjuntos en el segundo argumento, del operador $\otimes$ en el primer argumento, así como de cualquier operador en el cuerpo de las reglas. Finalmente, exigiendo la continuidad de $T_{\mathbb{P}}^{M}$, podemos asegurar que el menor modelo del programa puede ser obtenido computacionalmente. Todo esto se ilustra con un ejemplo en [122], donde se enfatiza la relevancia y las implicaciones del concepto de continuidad, especialmente si trabajamos en un entorno con errores en los datos. En primer lugar, se estudia el caso con operadores que son continuos y que aseguran la continuidad del operador de consecuencias inmediatas cuantificado, mientras que, posteriormente, se consideran unas t-normas que no son continuas, dando lugar, por lo tanto, a un operador que no es continuo, de manera que pequeñas variaciones en uno de los hechos, que se obtienen a través de observaciones, provocan un cambio drástico en el resultado final de la iteración.

2.3. Técnica computacional para la obtención de consecuencias en programación lógica multiadjunta con cuantificadores generalizados

En el artículo *Computational technique for obtaining consequences in multiadjoint logic programming with generalized quantifiers* [131], publicado en 2024 en la revista *Applied and Computational Mathematics*, cuyo factor de impacto

lo sitúa en el primer cuartil (Q1) de su categoría del *JCR*, presentamos una nueva semántica operacional, englobada en un entorno multiadjunto de lógica de proposiciones, para el marco flexible introducido en [129, 122], donde se trabaja con una generalización del operador de consecuencias inmediatas basada en el uso de cuantificadores generalizados determinados por medidas difusas.

El objetivo consistirá en establecer un método de razonamiento automático que permita derivar conclusiones del sistema modelado y responder a preguntas formuladas al programa. Para ello, como motor de inferencia, utilizaremos unas reglas admisibles, presentadas en [125], que permiten obtener nuevas fórmulas a partir de las reglas del programa lógico multiadjunto. Por lo tanto, la semántica operacional se fundamenta en la aplicación sucesiva de estas reglas admisibles a las reglas del programa hasta obtener una fórmula extendida sin símbolos proposicionales, cuya interpretación como un elemento del retículo en el que estemos trabajando es la respuesta calculada a la pregunta formulada al sistema. Esta nueva semántica operacional es de gran utilidad en una variedad de aplicaciones, entre las que destacan, los sistemas de predicción de tendencias y comportamientos, los sistemas de recomendación y de apoyo a la toma de decisiones, así como los sistemas de detección de fallos.

Por la posible presencia de ruido en las bases de datos, la definición tradicional de modelo de un programa se modificó, dando lugar al concepto de M-modelo, que depende de la familia de medidas difusas M que se considera [129]. En consecuencia, en este nuevo marco de trabajo más versátil y adaptable en el cual estamos trabajando, se modifica ligeramente la definición de respuesta correcta, introduciendo la noción de M-respuesta correcta, cuya definición dependerá de los M-modelos del programa. Entre todas estas M-respuestas correctas nos interesa la que más información aporta, siendo la máxima de todas. En [131, Teorema 6] se demuestra que la M-respuesta correcta máxima se corresponde con el menor punto fi-

jo del M-operador de consecuencias inmediatas cuantificado, $T_{\mathbb{P}}^{\mathrm{M}}$ y, por lo tanto, dicha respuesta puede ser obtenida computacionalmente a partir de este operador, siguiendo lo establecido en [129, Teorema 8]. Por esta razón, como se señala en [122], es de suma importancia que el operador de consecuencias inmediatas cuantificado sea continuo, con el fin de asegurar que su punto fijo y, por consiguiente, la M-respuesta correcta máxima, puedan alcanzarse.

No obstante, en un retículo general donde no se puede asegurar que la relación de orden sea total, puede ocurrir que la M-respuesta correcta máxima no pueda ser calculada a través de un proceso de razonamiento deductivo utilizando las reglas admisibles. Además, es más eficiente computacionalmente tener todas las reglas con la misma cabeza agrupadas en una sola. Por tales razones, se introduce la noción de M-reductante, que es una generalización del concepto de reductante dado en un entorno multiadjunto en [125]. Asimismo, esta extensión del concepto de reductante proporciona una guía para orientar futuras generalizaciones de este ambiente flexible a otros marcos de trabajo. La nueva definición de reductante se da a continuación. El detalle de los operadores y símbolos que se utilizan se puede consultar directamente en el artículo [131] o en el Capítulo 5.

Definición 2.2. *Sea* $M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi, \mathbb{P}_A \neq \varnothing\}$ *una familia de medidas difusas invariantes con respecto a la cardinalidad. Supongamos que todas las reglas en* $\mathbb{P}$ *con cabeza* A *son* $\langle A \leftarrow_{j_i} \mathcal{B}_i, \vartheta_i \rangle$*, para* $i \in \{1, \ldots, l\}$*, donde algunas reglas pueden ser hechos, y en consecuencia algunas fórmulas* $\mathcal{B}_i$ *pueden ser el símbolo* $\top$*. Un* M-reductante *para* A *es una regla* $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_l), \vartheta \rangle$ *tal que, para cualquier* $b_1, \ldots, b_l \in [0,1]$*, se satisface que*

$$Q_{\mu_A}(C_b^A) \;=\; \vartheta \dot{\&} \dot{@}\,(b_1, \ldots, b_l)$$

donde $(\leftarrow, \&)$ *es un par adjunto y la aplicación* $C_b^A \colon \mathbb{P} \to [0,1]$*, con* $b = (b_1, \ldots, b_l)$*,*

se define como

$$C_b^A\left(\langle D \leftarrow_i \mathcal{B}, \vartheta' \rangle\right) = \begin{cases} \vartheta' \dot{\&}_i b_i & \textit{si } D = A \\ 0 & \textit{en caso contrario} \end{cases}$$

Aunque el M-reductante para un símbolo proposicional sea un hecho, no todas las reglas del programa con cabeza el símbolo proposicional tienen que ser necesariamente hechos. De igual forma, para un símbolo proposicional determinado, puede existir una gran variedad de diferentes M-reductantes asociados a dicho símbolo proposicional, cambiando, por ejemplo, el operador de agregación considerado en el cuerpo del M-reductante. De entre todos los posibles M-reductantes que hay para un símbolo proposicional, vamos a considerar un tipo particular que tiene como valor de verdad el elemento máximo del retículo, al que nos referimos con el sobrenombre de canónico, y que siempre es posible construir para cualquier símbolo proposicional. Los M-reductantes canónicos son únicos excepto por el par adjunto que estemos considerando en su construcción, que a efectos de cálculo no afecta por tener el elemento máximo como valor de verdad.

Una de las primeras implicaciones que se derivan de esta nueva definición de reductante es que estas reglas también son satisfechas por los M-modelos del programa. En consecuencia, el conjunto de modelos del programa no varía, pudiendo considerarse una versión simplificada del programa original que tenga en cuenta únicamente un M-reductante para cada símbolo proposicional. Al programa obtenido se le denomina programa reducido por M-reductantes de $\mathbb{P}$, o simplemente r-programa. Esta definición también puede extenderse al caso del uso de M-reductantes canónicos, y teniendo en cuenta la unicidad de estos reductantes, podemos hacer referencia al r-programa canónico de un programa lógico. La justificación para poder trabajar con este programa reducido sin perder información del programa original radica en el hecho de que ambos son

semánticamente equivalentes, conforme a lo establecido en [131, Proposición 2] y [131, Proposición 4].

Posteriormente, en [131, Ejemplo 9] se prueba que hay respuestas calculadas que no son M-respuestas correctas. Esto se debe al hecho de que el concepto de respuesta correcta se ha flexibilizado, pudiendo haber respuestas calculadas que, efectivamente, no sean M-respuestas correctas. Por esta razón, resulta necesario modificar la noción de respuesta calculada para capturar la filosofía del uso de los cuantificadores generalizados. Introducimos el concepto de $\mathbb{P}_r$-respuesta calculada, dependiente del programa reducido $\mathbb{P}_r$ considerado y, por extensión, de la familia de medidas difusas M, como una respuesta calculada con respecto al programa reducido $\mathbb{P}_r$. La corrección del nuevo procedimiento, es decir, que para una pregunta $?A$ dada, todas las $\mathbb{P}_r$-respuestas calculadas son M-respuestas correctas, se presenta a continuación, aunque también se puede consultar en [131, Teorema 7].

Teorema 2.3. *Sea $\mathbb{P}_r$ un r-programa de $\mathbb{P}$, $?A$ una pregunta y M una familia de medidas difusas invariantes con respecto a la cardinalidad. Entonces, cada $\mathbb{P}_r$-respuesta calculada para $?A$ es una M-respuesta correcta para $\mathbb{P}$ y $?A$.*

Demostración. Sea λ una $\mathbb{P}_r$-respuesta calculada para la pregunta $?A$. Vamos a probar que λ es una M-respuesta correcta para $?A$ por inducción sobre la longitud n de la secuencia de cálculo de λ.

Para $n = 1$, se ha aplicado la regla admisible R2 o R3.

1. Si se ha aplicado R2, entonces el símbolo proposicional A no es la cabeza de ninguna regla o hecho en $\mathbb{P}_r$. La correspondiente secuencia de fórmulas extendidas es $G_0 = A$, $G_1 = \bot$, de donde $\lambda = \bot$, que siempre es una M-respuesta correcta.

2. Si se ha aplicado R3, entonces hay un hecho $\langle A \leftarrow \top, \vartheta \rangle \in \mathbb{P}_r$, donde $\vartheta = Q_{\mu_A}\left(C_b^A\right)$, y la correspondiente secuencia de fórmulas extendi-

das es $G_0 = A$, $G_1 = \vartheta$. Por lo tanto, $\lambda = \vartheta$. Ahora, de la definición de M-modelo, si $I \in \mathcal{I}_{\mathfrak{L}}$ es un M-modelo, entonces $T_{\mathbb{P}}^{\mathrm{M}}(I) \sqsubseteq I$ y, en consecuencia, $\lambda = \vartheta = Q_{\mu_A}\left(C_b^A\right) = T_{\mathbb{P}}^{Q_{\mu_A}}(I)(A) \leq I(A)$, donde $b = (b_1, \ldots, b_{|\mathbb{P}_A|}) = (\hat{I}(\mathcal{B}_1), \ldots, \hat{I}(\mathcal{B}_{|\mathbb{P}_A|}))$. Luego, λ es una M-respuesta correcta en $\mathbb{P}_r$.

Supongamos que el resultado se verifica para una longitud de la secuencia de cálculo de λ menor que n, donde $1 < n$, es decir, si λ es una $\mathbb{P}_r$-respuesta calculada para $?A$ con una longitud de cálculo menor que n, entonces λ es una M-respuesta correcta para $?A$. En este contexto, debe existir necesariamente alguna regla con cabeza A en $\mathbb{P}_r$, ya que en caso contrario la longitud de la secuencia de cálculo de λ sería 1. Por tanto, podemos construir el M-reductante $\langle A \leftarrow @\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right), \vartheta \rangle$, donde $@\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right) = @'\left[B_1, \ldots, B_k\right]$, en el cual $@'$ es el agregador definido como la composición de todos los símbolos de operadores en $@\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right)$ y $B_1, \ldots, B_k$ son los símbolos proposicionales que aparecen en el cuerpo del M-reductante. Obtenemos la secuencia de fórmulas extendidas

$$G_0 = A, G_1 = \vartheta \bar{\&} \bar{@}'\left[B_1, \ldots, B_k\right], \ldots, G_r = \vartheta \bar{\&} \bar{@}'\left[\lambda_1, \ldots, \lambda_k\right] = \lambda$$

donde la secuencia de cálculo para cada λ_i tiene una longitud menor que n y, por la hipótesis de inducción, se verifica que $\lambda_i \leq I(B_i)$ para todo M-modelo I, e $i \in \{1, \ldots, k\}$. En consecuencia, llegamos a la siguiente cadena de desigualdades.

$$\begin{aligned}
\lambda &= \vartheta \bar{\&} \dot{@}'\left[\lambda_1, \ldots, \lambda_k\right] \\
&\overset{(1)}{\leq} \vartheta \bar{\&} \dot{@}'\left[I(B_1), \ldots, I(B_k)\right] \\
&= \vartheta \bar{\&} \hat{I}\left(@'\left[B_1, \ldots, B_k\right]\right) \\
&= \vartheta \bar{\&} \hat{I}\left(@\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right)\right) \\
&= \vartheta \bar{\&} \dot{@}\left(\hat{I}(\mathcal{B}_1), \ldots, \hat{I}(\mathcal{B}_l)\right)
\end{aligned}$$

$$\begin{aligned} &\overset{(2)}{=} Q_{\mu_A}\left(C_I^A\right) \\ &= T_{\mathbb{P}}^{\mathrm{M}}(I)(A) \\ &\overset{(3)}{\leq} I(A) \end{aligned}$$

donde (1) se verifica por la monotonía de $\dot{@}$ y $\bar{\&}$, y por la hipótesis de inducción $\lambda_i \leq I(B_i)$, para todo $i \in \{1, \ldots, k\}$, (2) es consecuencia de la definición de M-reductante, mientras que (3) resulta del hecho de que I es un M-modelo. Por lo tanto, λ es una M-respuesta correcta para la pregunta $?A$. □

A continuación, estudiamos algunos resultados de completitud, cuyo objetivo es demostrar que las M-respuestas correctas pueden ser derivadas dentro del sistema utilizando las reglas de inferencia, esto es, que pueden determinarse a través de $\mathbb{P}_r$-respuestas calculadas. En primer lugar, se precisa un resultado previo que establece que cualquier iteración del M-operador de consecuencias inmediatas cuantificado sobre un símbolo proposicional A es una $\mathbb{P}_r$-respuesta calculada para $?A$. Para los siguientes resultados se necesita que el operador de consecuencias inmediatas cuantificado sea continuo. El primero de los resultados de completitud [131, Teorema 8], que realmente es de cuasicompletitud, establece que cualquier M-respuesta correcta puede ser aproximada por una $\mathbb{P}_r$-respuesta calculada. El siguiente resultado [131, Teorema 9], que también es de cuasicompletitud, afirma que para cada M-respuesta correcta existe una secuencia de elementos de tal forma que su supremo es mayor que la M-respuesta correcta y para cualquier elemento de esta secuencia, siempre va a existir una $\mathbb{P}_r$-respuesta calculada mayor que este.

Por último, estudiamos el caso de la $\mathbb{P}_r$-respuesta calculada máxima, que se define como la mayor de todas las $\mathbb{P}_r$-respuestas calculadas. Antes de todo, observemos que puede ocurrir que esta $\mathbb{P}_r$-respuesta calculada máxima no exista si, al aplicar las reglas admisibles en el proceso de inferencia, obtenemos una cadena de $\mathbb{P}_r$-respuestas calculadas creciente sin

máximo, como se muestra en [131, Ejemplo 10]. No obstante, esto no supone un inconveniente significativo puesto que los resultados previos sobre cuasicompletitud demuestran que es posible aproximarnos tanto como queramos a la $\mathbb{P}_r$-respuesta calculada máxima. Además, como consecuencia de [131, Proposición 6], se puede establecer una relación entre las iteraciones del $T_{\mathbb{P}}^{\mathrm{M}}$ y las $\mathbb{P}_r$-respuestas calculadas. La no existencia de una $\mathbb{P}_r$-respuesta calculada máxima, debido a su imposibilidad de cálculo, está vinculada con la no terminación del M-operador de consecuencias inmediatas cuantificado.

En el supuesto de que esta $\mathbb{P}_r$-respuesta calculada máxima exista, en [131, Teorema 10] presentamos un procedimiento para poder calcularla, utilizando para ello los M-reductantes canónicos y, por extensión, el programa reducido. Sin embargo, nótese que todavía no podemos precisar cuál es esta $\mathbb{P}_r$-respuesta calculada máxima. En [131, Teorema 11] se establece que la $\mathbb{P}_r$-respuesta calculada máxima es el menor punto fijo del M-operador de consecuencias inmediatas cuantificado, dando una forma de calcularlo automáticamente a partir de este operador. Este hecho nos permite concluir en [131, Corolario 1] que la $\mathbb{P}_r$-respuesta calculada máxima coincide con la M-respuesta correcta máxima. Por lo tanto, con la introducción de esta nueva versión del reductante y del programa reducido, hemos podido calcular computacionalmente la M-respuesta correcta máxima. Cabe destacar la importancia de que el M-operador de consecuencias inmediatas cuantificado sea continuo, ya que en caso contrario no podemos asegurar que esta $\mathbb{P}_r$-respuesta calculada máxima se alcance.

2.4. Operador de consecuencias inmediatas a través de operadores de promedio ponderado ordenado

En los tres artículos anteriores [129, 122, 131] hemos presentado una definición alternativa del operador de consecuencias inmediatas basada en cuantificadores generalizados, además de haber estudiado tanto una semántica de punto fijo como una semántica operacional para la nueva aproximación. El operador de consecuencias inmediatas cuantificado verifica propiedades fundamentales como la continuidad, que lo convierten en una herramienta indispensable en la programación lógica para el tratamiento del ruido en las bases de datos.

Sin embargo, la flexibilidad del nuevo operador resulta más limitada cuando es necesario abordar, por ejemplo, situaciones donde la agregación de los datos deba tener en cuenta las relaciones de importancia que se establecen entre los diferentes elementos. En otras palabras, puede no capturar adecuadamente el peso específico que se le asigna a cada uno de los elementos. Teniendo en cuenta el contexto de introducción de los operadores de promedio ponderado ordenado, operadores OWA para simplificar, en la teoría de decisión multicriterio, de lo que hemos tratado previamente en la Sección 1.1.9, en el artículo *Immediate consequences operator through ordered weighted average operators* [130], publicado en 2023 en la revista *Iranian Journal of Fuzzy Systems*, situada en el primer cuartil (Q1) de su categoría del *JCR*, presentamos una nueva definición del operador de consecuencias inmediatas basada en estos operadores, con el objetivo de reducir el impacto de la posible presencia de información errónea en las bases de datos que modelan las reglas lógicas de los programas. Todo ello en un entorno de programación lógica multiadjunta finita, siendo este último requisito indispensable ya que estamos trabajando con operadores de

agregación con un número finito de argumentos.

Entre las aportaciones más relevantes del presente artículo destaca la propuesta de una nueva definición del operador de consecuencias inmediatas basada en los operadores OWA. Específicamente, la nueva definición se da a continuación. El detalle de los operadores y los símbolos que se utilizan se puede consultar directamente en el artículo [130] o en el Capítulo 6.

Definición 2.3. *Sea* $W = (w_1, \ldots, w_n) \in [0,1]^n$ *un vector de pesos, tal que verifica* $\sum_{i=1}^{n} w_i = 1$. *El* operador de agregación ponderado de consecuencias inmediatas $T_{\mathbb{P}}^{W} : \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ *está definido, para* $I \in \mathcal{I}_{\mathfrak{L}}$ *y* $A \in \Pi$, *como*

$$T_{\mathbb{P}}^{W}(I)(A) \quad = \quad @_W \left(C_I^A(u_1), \ldots, C_I^A(u_n) \right) \tag{2.3}$$

donde $\{u_1, \ldots, u_n\}$ *es el conjunto de reglas de* $\mathbb{P}$ *y la aplicación* $C_I^A \colon \mathbb{P} \longrightarrow [0,1]$ *se define, para cada* $\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}$, *como*

$$C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle\right) = \begin{cases} \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) & \text{si } C = A \\ 0 & \text{en caso contrario} \end{cases}$$

En concreto, dada una interpretación I y un símbolo proposicional A, el operador de agregación ponderado de consecuencias inmediatas asociado al vector de pesos W, $T_{\mathbb{P}}^{W}$, se define como la agregación de los valores aportados por cada regla de $\mathbb{P}$, que se obtienen a través de la aplicación C_I^A, que también fue considerada anteriormente para el caso de los cuantificadores generalizados [129]. Los pesos asociados en la agregación dependen del vector de pesos W, de tal forma que el peso que ocupa la primera posición del vector está asociado con el valor más alto de la agregación; el peso de la segunda posición está asociado con el segundo valor más alto, y así sucesivamente. En [130, Proposición 3.3.] se prueba que, efectivamente, esta definición generaliza el caso clásico dado en un entorno multiadjunto cuando se considera el vector de pesos asociado al operador máximo.

Una de las primeras propiedades que esta nueva versión más flexible del operador de consecuencias inmediatas satisface es la monotonía, como se establece en [130, Proposición 3.4.].

Con el objetivo de introducir una semántica de punto fijo adaptada a este entorno más versátil, estudiamos un caso particular del operador de agregación ponderado de consecuencias inmediatas. Si en el contexto de los cuantificadores nos quedamos con el segundo valor más alto para solucionar la posible presencia de ruido en los datos, en este caso, con la intención de evitar la pérdida de información que supone eliminar el valor más alto, vamos a seleccionar un vector de pesos con el que sea posible quedarnos con la media de los dos valores más altos, esto es, $W_2 = \left(\frac{1}{2}, \frac{1}{2}, 0, \overset{(n-2)}{\ldots}, 0\right)$, siendo n el número de reglas de $\mathbb{P}$. Considerando que estamos operando en un nuevo ambiente, la definición de modelo tiene que adaptarse y, en consecuencia, se introduce la noción de W_2-modelo.

Es importante señalar que tendremos tantas definiciones diferentes de modelos como vectores de pesos se elijan para debilitar el carácter estricto del supremo y, por tanto, dando lugar a un nuevo ambiente de trabajo. En este caso, como el operador de agregación ponderado de consecuencias inmediatas asociado al vector de pesos W_2 se queda con una media de los dos valores más altos, diremos que una interpretación I es un W_2-modelo si todas las reglas del programa son satisfechas por I excepto posiblemente una de ellas, en cuyo caso la media de los dos valores más altos aportados por las reglas, es decir, $\vartheta\dot{\&}_i\hat{I}(\mathcal{B})$, es menor que el valor de la interpretación en el símbolo proposicional A. La idea subyacente a esta definición es la necesidad de buscar que nuestra interpretación satisfaga la descripción lingüística asociada al vector de pesos, como ya lo hiciera anteriormente con la descripción lingüística asociada al cuantificador generalizado o, de manera más precisa, a la medida difusa [129].

El concepto de W_2-modelo satisface la caracterización de los modelos como puntos posfijos del operador de agregación ponderado de conse-

cuencias inmediatas asociado al vector de pesos W_2, como demuestra el siguiente resultado.

Teorema 2.4. *Una interpretación I es un W_2-modelo para $\mathbb{P}$ si y solo si $T_{\mathbb{P}}^{W_2}(I) \sqsubseteq I$.*

Demostración. Sean $\{u_1, \ldots, u_n\}$ el conjunto de reglas de $\mathbb{P}$, y π la permutación considerada en la definición del operador OWA [130, Definición 2.27.].

Si todas las reglas de $\mathbb{P}$ son satisfechas por I, entonces en particular se cumple que $C_I^A\left(u_{\pi(1)}\right)$, $C_I^A\left(u_{\pi(2)}\right) \leq I(A)$ y, por lo tanto

$$T_{\mathbb{P}}^{W_2}(I)(A) = \frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A)$$

Por otro lado, si alguna regla no es satisfecha por I, entonces necesariamente debe ser $u_{\pi(1)}$. Como I es un W_2-modelo, de la definición de W_2-modelo, se satisface que

$$T_{\mathbb{P}}^{W_2}(I)(A) = \frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A)$$

A continuación, para probar la implicación recíproca, supongamos que la interpretación I verifica que $T_{\mathbb{P}}^{W_2}(I) \sqsubseteq I$. Teniendo en cuenta la permutación π, se satisface que $C_I^A\left(u_{\pi(2)}\right) \leq C_I^A\left(u_{\pi(1)}\right)$, de donde llegamos a la siguiente expresión

$$C_I^A\left(u_{\pi(n)}\right) \leq \ldots \leq C_I^A\left(u_{\pi(2)}\right) \leq \frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \overset{(*)}{\leq} I(A)$$

donde (*) es consecuencia de la definición de $T_{\mathbb{P}}^{W_2}(I)$ y de la expresión $T_{\mathbb{P}}^{W_2}(I) \sqsubseteq I$. En consecuencia, todas las reglas de $\mathbb{P}$ son satisfechas por I excepto posiblemente $u_{\pi(1)}$, en cuyo caso se cumple que $\frac{C_I^A\left(u_{\pi(1)}\right)+C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A)$. □

Considerando la monotonía del $T_{\mathbb{P}}^{W_2}$ y la caracterización de los W_2-modelos como puntos posfijos del $T_{\mathbb{P}}^{W_2}$, podemos identificar el menor W_2-modelo del programa con el menor punto fijo del operador de agregación

ponderado de consecuencias inmediatas asociado a W_2. Este hecho permite trasladar la dificultad de determinar el significado de un programa lógico con la ejecución de un algoritmo para obtener deducciones automáticas, que una máquina puede realizar de manera independiente, basado en el operador de agregación ponderado de consecuencias inmediatas. De hecho, el último resultado de este caso particular, [130, Teorema 3.9.], establece que es posible obtener el menor W_2-modelo iterando el operador de agregación ponderado de consecuencias inmediatas un número ordinal de veces α, partiendo de la menor interpretación $\triangle$.

A partir de este ejemplo específico podemos constatar que no es tan sencillo establecer una definición general de modelo para un vector de pesos arbitrario, considerando que, al no conocer el vector de pesos particular con el que estemos trabajando, resulta complicado proporcionar su descripción lingüística. Dado que para que todo el proceso funcione es necesario obtener la ya mencionada caracterización de los modelos como los puntos posfijos del operador, definiremos los W-modelos, dependientes del vector de pesos W, como los puntos posfijos del $T_{\mathbb{P}}^{W}$.

Si bien esta definición contribuye a eliminar la posible presencia de ruido en los datos, todavía tenemos el problema asociado a aquellos símbolos que son cabeza de una única regla. Si consideramos el vector de pesos $W = (0, 1, 0, \overset{(n-2)}{\ldots}, 0)$, donde n es el número de reglas del programa, y calculamos el valor del $T_{\mathbb{P}}^{W}$ asociado a los símbolos proposicionales que sean cabeza de una única regla, se obtiene el valor 0, ya que no existe un segundo valor más alto asociado con esa regla. Por lo tanto, perderíamos toda la información ofrecida por dicha regla. Una posible solución consiste en considerar un vector de pesos más flexible, que dependa del número de reglas del programa vinculadas a cada símbolo proposicional.

$$\tau(A) = \begin{cases} 1 & \text{si} \quad |\mathbb{P}_A| = 1 \\ (\alpha_1, \alpha_2) & \text{si} \quad |\mathbb{P}_A| = 2, \quad \text{con} \quad \alpha_1 + \alpha_2 = 1 \\ \left(\alpha_1, \alpha_2, \alpha_3, \overset{(n)}{\ldots}, \alpha_n\right) & \text{si} \quad |\mathbb{P}_A| = n \geq 3, \\ & \qquad \text{con} \quad \alpha_1 + \ldots + \alpha_n = 1 \end{cases}$$

Esta idea en la que podemos seleccionar el vector de pesos más apropiado para cada símbolo proposicional se puede extender a considerar una familia de vectores de pesos

$$\mathcal{W} = \left\{ W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1, A \in \Pi, \mathbb{P}_A \neq \varnothing, |\mathbb{P}_A| = n_A \right\}$$

Para ello, presentamos una generalización de la definición del operador de agregación ponderado de consecuencias inmediatas asociado a la familia $\mathcal{W}$ [130, Definición 3.11.]. Además, en efecto, se trata de una generalización ya que, dado un vector de pesos W arbitrario, siempre es posible obtener una familia de vectores de pesos $\mathcal{W}$ de manera que el operador de consecuencias inmediatas asociado a $\mathcal{W}$ sea igual al $T_{\mathbb{P}}^{W}$. Esta es la razón por la que estamos considerando vectores de pesos W_A menos restrictivos, verificando que $\sum_{i=1}^{n_A} W_A[i] \leq 1$. Asimismo, esta nueva definición satisface la propiedad de monotonía.

Con respecto a la semántica de punto fijo de esta aproximación, debido a la complejidad de proveer una descripción lingüística a cada vector de pesos de la familia, definiremos los $\mathcal{W}$-modelos, dependientes de la familia de vectores de pesos $\mathcal{W}$, como los puntos posfijos de $T_{\mathbb{P}}^{\mathcal{W}}$. Nuevamente, en este caso es posible definir el menor $\mathcal{W}$-modelo como el menor punto fijo del operador de agregación ponderado de consecuencias inmediatas asociado a $\mathcal{W}$ [130, Proposición 3.15.], a la vez que es posible calcularlo

mediante un procedimiento computacional basado en este operador [130, Teorema 3.16.].

Podemos concluir que en [130] hemos presentado una generalización del operador de consecuencias inmediatas, que depende del vector de pesos seleccionado, aportando una mayor versatilidad en la modelización de sistemas expertos, por ejemplo. Además, el nuevo enfoque permite una mejor adaptación a diferentes escenarios y requisitos específicos. No obstante, queda pendiente la complicada tarea de seleccionar una familia de vectores de pesos adecuada. De manera más bien didáctica, si elegimos el vector de pesos mínimo $W_* = (0, 0, \ldots, 0, 1)$, se obtiene la definición opuesta del operador de consecuencias inmediatas clásico. No obstante, esto no nos interesa; en su lugar resulta más conveniente considerar valores cercanos al supremo. Para ello, necesitamos introducir una medida que nos asegure que elegimos vectores de pesos próximos al máximo, que es aquel con el que se obtiene la definición del operador de consecuencias inmediatas basada en el supremo. Con este objetivo, y basándonos en el índice de optimismo introducido por Ronald Yager en [209], introducimos el concepto de índice de optimismo asociado a la familia de vectores de pesos $\mathcal{W}$ como el ínfimo de los índices de optimismo de cada uno de los vectores de pesos de dicha familia. De esta forma, el usuario dispone de un indicador para poder seleccionar una familia de vectores de pesos adecuada para poder eliminar el ruido presente en la base de datos, a la vez que intenta preservar la definición original del operador de consecuencias inmediatas.

Hemos comentado que en [130, Teorema 3.16.] se prueba que existe un ordinal α tal que el menor punto fijo de $T_{\mathbb{P}}^{\mathcal{W}}$ y, por consiguiente el menor $\mathcal{W}$-modelo del programa, es $\left(T_{\mathbb{P}}^{\mathcal{W}}\right)^{\alpha}(\triangle)$. No obstante, tal como se ha expuesto anteriormente en [122], puede suceder que nunca se alcance el menor punto fijo de $T_{\mathbb{P}}^{\mathcal{W}}$, por lo que la exigencia de la continuidad del operador se convierte en una prioridad fundamental. En [130] se estudian

condiciones suficientes y necesarias para poder asegurar la continuidad del nuevo operador. El siguiente resultado establece las condiciones suficientes.

Teorema 2.5. *Si todos los operadores presentes en el cuerpo de las reglas del programa $\mathbb{P}$ son continuos Scott, y los conjuntores adjuntos también son continuos Scott en el segundo argumento, entonces $T_{\mathbb{P}}^{\mathcal{W}}$ es continuo Scott.*

Demostración. Tenemos que probar, para cada subconjunto dirigido de interpretaciones X y cada símbolo proposicional A, que

$$T_{\mathbb{P}}^{\mathcal{W}}(\sup X)(A) = \sup\left\{T_{\mathbb{P}}^{\mathcal{W}}(J)(A) \mid J \in X\right\} \tag{2.4}$$

Denotaremos $S = \sup X$. Si $\mathbb{P}_A = \varnothing$, entonces $T_{\mathbb{P}}^{\mathcal{W}}(\sup X)(A) = 0 = \sup\left\{T_{\mathbb{P}}^{\mathcal{W}}(J)(A) \mid J \in X\right\}$. En caso contrario, para $\mathbb{P}_A \neq \varnothing$, se obtiene la siguiente cadenas de igualdades.

$$\begin{aligned}
T_{\mathbb{P}}^{\mathcal{W}}(\sup X)(A) &= T_{\mathbb{P}}^{W_A}(\sup X)(A) \\
&= \sum_{i=1}^{n_A} W_A[i]\, C_S^A\left(u_{\pi_A(i)}\right) \\
&= \sum_{i=1}^{n_A} W_A[i]\, \left(\vartheta_{\pi_A(i)} \dot{\&}_{\pi_A(i)} \widehat{S}\left(\mathcal{B}_{\pi_A(i)}\right)\right) \\
&\overset{(1)}{=} \sum_{i=1}^{n_A} W_A[i]\, \left(\vartheta_{\pi_A(i)} \dot{\&}_{\pi_A(i)} \sup\left\{\hat{J}(\mathcal{B}_{\pi_A(i)}) \mid J \in X\right\}\right) \\
&\overset{(2)}{=} \sum_{i=1}^{n_A} W_A[i]\, \sup\left\{\vartheta_{\pi_A(i)} \dot{\&}_{\pi_A(i)} \hat{J}(\mathcal{B}_{\pi_A(i)}) \mid J \in X\right\} \\
&\overset{(3)}{=} \sup\left\{\sum_{i=1}^{n_A} W_A[i]\, \left(\vartheta_{\pi_A(i)} \dot{\&}_{\pi_A(i)} \hat{J}(\mathcal{B}_{\pi_A(i)})\right) \mid J \in X\right\} \\
&= \sup\left\{\sum_{i=1}^{n_A} W_A[i]\, C_J^A\left(u_{\pi(i)}\right) \mid J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{W_A}(J)(A) \,|\, J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{\mathcal{W}}(J)(A) \,|\, J \in X\right\}
\end{aligned}$$

donde la igualdad (1) se verifica por [130, Lema 2.26.], la igualdad (2) es consecuencia de la continuidad de Scott de los operadores $\&_i$ en el segundo argumento, mientras que (3) se deduce del hecho de que el producto y la suma son aplicaciones continuas y, por tanto, continuas Scott en el intervalo unidad, en particular. Por lo tanto, podemos concluir que la nueva definición del operador de consecuencias inmediatas, $T_{\mathbb{P}}^{\mathcal{W}}$, es continua Scott. □

Las condiciones necesarias de la continuidad se establecen en [130, Teorema 4.3.], estableciendo que la continuidad de los conjuntores adjuntos es necesaria para poder asegurar la continuidad del $\mathcal{W}$-operador de agregación ponderado de consecuencias inmediatas. Finalmente, si nuestro operador es continuo, podemos asegurar que el menor $\mathcal{W}$-modelo se obtiene en a lo sumo una cantidad numerable de iteraciones del $T_{\mathbb{P}}^{\mathcal{W}}$ [130, Corolario 4.4.]. Todo esto hace que el operador de agregación ponderado de consecuencias inmediatas sea una herramienta muy poderosa para el tratamiento de la incertidumbre en las bases de datos.

2.5. Semántica operacional sobre programación lógica de primer orden, difusa y arbitraria

La continuación lógica del artículo anterior *Immediate consequences operator through ordered weighted average operators* [130], tras haber estudiado una semántica de punto fijo, sería complementarla con una semántica operacional. Puesto que, para cada semántica de punto fijo basada en una modificación del operador de consecuencias inmediatas, resulta necesario estudiar la semántica operacional, pensamos que sería más conveniente realizar este estudio sobre una aproximación de la programación lógica multiadjunta basada en un operador de consecuencias inmediatas monótono y arbitrario $T_{\mathbb{P}}^{\diamond}$, abarcando así las dos extensiones presentadas en los

artículos previos. Además, para aumentar el poder expresivo, se ha planteado con respecto a la lógica de primer orden. Por lo tanto, en este trabajo, enviado para publicar a una de las revistas más prestigiosas del área, presentamos una semántica operacional válida para cualquier entorno de programación lógica multiadjunta de primer orden, la cual está basada en una definición monótona del operador de consecuencias inmediatas. Además, todo el estudio se realiza con respecto a un retículo general $\langle L, \preceq \rangle$.

Sin perder de vista tampoco el objetivo de introducir una semántica operacional para la aproximación de la programación lógica con el uso de los operadores de promedio ponderado ordenado [130], en este artículo todos los ejemplos particulares analizados están basados en dicha extensión del operador de consecuencias inmediatas, adaptado a la lógica de primer orden.

En primer lugar, se definen las reglas y pasos admisibles, junto con el paso interpretativo [97], que conforman la base estructural de la semántica operacional, y establecen los pasos necesarios que permiten derivar conclusiones de una manera coherente y correcta. En este nuevo ambiente más flexible en el que estamos trabajando, es preciso aportar una nueva definición de respuesta correcta. Específicamente, se introduce el concepto de $\diamond$-respuesta correcta difusa, basada en la noción de $\diamond$-modelo, que serán aquellas interpretaciones que satisfacen la descripción lingüística asociada a $\diamond$. Para facilitar la comprensión, ilustramos la definición de $\diamond$-modelo con el caso particular de los cuantificadores generalizados [131].

Para una sustitución θ fija, la $\diamond$-respuesta correcta difusa que más información aporta asociada a dicha sustitución es la θ-$\diamond$-respuesta correcta máxima. En [128, Teorema 6] se demuestra que, para una pregunta $\mathcal{Q}$ y una sustitución θ, existe un ordinal α tal que $\left(\widehat{(T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle)}(\mathcal{Q}\theta); \theta\right)$ es la θ-$\diamond$-respuesta correcta máxima. Como el operador $T_{\mathbb{P}}^{\diamond}$ es monótono y es un endomorfismo entre retículos completos, por el Teorema de Punto Fijo de Tarski [118], siempre va a existir un ordinal α tal que $(T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle) = \mathrm{lfp}(T_{\mathbb{P}}^{\diamond})$.

En consecuencia, la θ-$\diamond$-respuesta correcta máxima se asocia con el menor punto fijo de $T_{\mathbb{P}}^{\diamond}$, por lo que se puede obtener computacionalmente realizando sucesivas iteraciones del $T_{\mathbb{P}}^{\diamond}$ hasta alcanzar la estabilidad.

Al tratarse de un retículo general, existen situaciones en las que la respuesta correcta que más información aporta, es decir, la θ-$\diamond$-respuesta correcta máxima, no puede obtenerse a través de una respuesta calculada, esto es, no se puede derivar a partir de las reglas admisibles. Esta es la razón por la que se introduce una nueva definición de reductante adaptada a este entorno. Específicamente, la definición de $\diamond$-reductante asociado a un conjunto de reglas críticas C se da a continuación.

Definición 2.4. *Sea* $C = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq r\}$, *con* $\theta = mgu\{H_1, \ldots, H_r\}$, *un conjunto de reglas críticas en* $\mathbb{P}$, *tal que* $|\mathcal{S}_g(\mathcal{B}_i\theta)| = n_i \in \mathbb{N}$, *para todo* $i \in \{1, \ldots, r\}$. *Se dice que la regla*

$$\big\langle H_1\theta\theta_1 \leftarrow @(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r)); \top\big\rangle$$

es un C-$\diamond$-reductante crítico *de* $\mathbb{P}$, *de manera breve* C-$\diamond$-*reductante o* $\diamond$-*reductante asociado a* C, *donde* $\theta_1 \in \mathcal{S}_g(H_1\theta)$, $\xi_1^i, \ldots, \xi_{n_i}^i \in \mathcal{S}_g(\mathcal{B}_i\theta\theta_1)$, *para todo* $i \in \{1, \ldots, r\}$, $\leftarrow$ *es cualquier implicación con un conjuntor adjunto,* $@_{\sup}$ *se define como*

$$\dot{@}_{\sup}(b_1, \ldots, b_{n_i}) = \sup\{b_1, \ldots, b_{n_i}\}$$

para cualquier $b_1, \ldots, b_{n_i} \in L$, *y la función de verdad para el símbolo* $@$ *es un operador de agregación débil que satisface*

$$T_{\mathbb{P}}^{\diamond}(I)(H_1\theta\theta_1) \;=\; \dot{@}\,(w_1, \ldots, w_r)$$

donde $w_i = \dot{@}_{\sup}(\vartheta_i \dot{\&}_i b_1^i, \ldots, \vartheta_i \dot{\&}_i b_{n_i}^i)$, *para todo* $i \in \{1, \ldots, r\}$, *e* I *es una interpretación tal que*

$$\hat{I}(\mathcal{B}_i\theta\theta_1\xi_1^i) = b_1^i, \ldots, \hat{I}(\mathcal{B}_i\theta\theta_1\xi_{n_i}^i) = b_{n_i}^i, \textit{ para todo } i \in \{1, \ldots, r\}$$

es decir,

$$S_I\left(\vartheta_i \&_i \mathcal{B}_i \theta\theta_1\right) = \dot{@}_{\sup}(\vartheta_i \dot{\&}_i b_1^i, \ldots, \vartheta_i \dot{\&}_i b_{n_i}^i) = w_i, \textit{ para todo } i \in \{1, \ldots, r\}$$

Además, se define el $\diamond$-reductante crítico máximo asociado a un átomo básico A*, brevemente* $\diamond$*-reductante para* A*, como el* $\diamond$*-reductante crítico asociado a* C_M^A*, donde* C_M^A *es el máximo conjunto de reglas críticas de* $\mathbb{P}$ *asociadas a* A*.*

Esta definición fue todo un reto y se puede considerar como una de las aportaciones más destacadas de este trabajo. Teniendo en cuenta que el $\diamond$-reductante para cada átomo básico es único, podemos considerar un nuevo programa lógico multiadjunto formado por el $\diamond$-reductante asociado a cada átomo básico que pueda unificar con la cabeza de una regla del programa original. Este programa, llamado programa formado por $\diamond$-reductantes de $\mathbb{P}$, o brevemente r-programa y que también fue considerado en [131], es semánticamente equivalente al original $\mathbb{P}$, como se demuestra en [128, Proposiciones 4 y 6]. Este nuevo programa no solo simplifica todo el procedimiento, sino que será fundamental para introducir un nuevo concepto de respuesta calculada adaptada a la posible presencia de ruido en las bases de datos. En particular, se define una $\mathbb{P}_r$-respuesta calculada admisible como una respuesta calculada admisible con respecto al programa reducido y a la misma pregunta, utilizando para su cálculo las reglas admisibles introducidas. De la misma forma, se define una $\mathbb{P}_r$-respuesta calculada difusa como una respuesta calculada difusa con respecto al programa reducido. La diferencia entre ambos tipos de respuestas radica en el hecho de que la primera ha sido obtenida a partir de los pasos admisibles, hasta llegar a una fórmula sin átomos; mientras que la segunda es la que se obtiene al aplicarle el paso interpretativo a la respuesta calculada admisible.

El siguiente paso natural consiste en estudiar la corrección del procedimiento planteado, es decir, en asegurar que cualquier $\mathbb{P}_r$-respuesta cal-

culada difusa es una $\diamond$-respuesta correcta. Todo ello se demuestra en [128, Teorema 7].

Teorema 2.6. *Sea A un átomo y $\mathbb{P}_r$ el r-programa de $\mathbb{P}$. Entonces, cada $\mathbb{P}_r$-respuesta calculada difusa $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ para $?A$ es una $\diamond$-respuesta correcta difusa $(\dot{\mathcal{A}}; \sigma)$ para $\mathbb{P}$ y $?A$, donde σ es el unificador más general (*mgu*) de A y la cabeza de una regla en $\mathbb{P}_r$.*

Demostración. Consideremos un átomo A y una $\mathbb{P}_r$-respuesta calculada difusa $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$. Vamos a probar el resultado por inducción sobre la longitud n de la secuencia de cálculo de la $\mathbb{P}_r$-respuesta calculada admisible $\langle \mathcal{A}; \sigma \rangle$, de la cual se obtiene la $\mathbb{P}_r$-respuesta calculada difusa $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ para $?A$, aplicando el paso interpretativo. En primer lugar, supongamos que se obtiene en un número de pasos $n = 1$. Para este caso, surgen las siguientes posibilidades.

1. El átomo A no unifica con la cabeza de ninguna regla en $\mathbb{P}_r$. La correspondiente secuencia para obtener $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ es

   $$\langle A; id \rangle \rightsquigarrow_{\text{AS}} \langle \bot; id \rangle \rightsquigarrow_{\text{IS}} \langle \bot; id \rangle$$

   que siempre es una $\diamond$-respuesta correcta difusa para $?A$.

2. El átomo A unifica con la cabeza $H_1\theta\theta_1$ de una regla en $\mathbb{P}_r$. En este caso, como $n = 1$, la regla de $\mathbb{P}_r$ proviene del conjunto de hechos críticos $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \top; \vartheta_i \rangle \in \mathbb{P} \mid 1 \leq i \leq r\}$, con $\theta = \text{mgu}\{H_1, \ldots, H_r\}$ y $\theta_1 \in \mathcal{S}_g(H_1\theta)$. Por tanto, la regla en $\mathbb{P}_r$ es

   $$\langle H_1\theta\theta_1 \leftarrow @(\vartheta_1, \ldots, \vartheta_r); \top \rangle$$

   y existe una sustitución ν tal que $A\nu = H_1\theta\theta_1$. La $\mathbb{P}_r$-respuesta calculada difusa se obtiene de la secuencia

   $$\langle A; id \rangle \rightsquigarrow_{\text{AS}} \langle @(\vartheta_1, \ldots, \vartheta_r); \nu \rangle \rightsquigarrow_{\text{IS}} \left\langle \dot{@}(\vartheta_1, \ldots, \vartheta_r); \nu \right\rangle$$

donde, de la definición de $\diamond$-reductante, se satisface que $\dot{@}(\vartheta_1, \ldots, \vartheta_r) = T^{\diamond}_{\mathbb{P}}(I)(H_1\theta\theta_1)$ con I cualquier interpretación. Por tanto, se puede considerar un $\diamond$-modelo I de $\mathbb{P}$ y entonces,

$$\dot{@}(\vartheta_1, \ldots, \vartheta_r) = T^{\diamond}_{\mathbb{P}}(I)(H_1\theta\theta_1) \preceq I(H_1\theta\theta_1) = I(A\nu)$$

En consecuencia, $\left\langle \dot{@}(\vartheta_1, \ldots, \vartheta_r); \nu \right\rangle$ es una $\diamond$-respuesta correcta difusa para $?A$.

Supongamos que el resultado es cierto para toda longitud de cálculo menor que n, con $1 < n$, es decir, si $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ es una $\mathbb{P}_r$-respuesta calculada difusa para una pregunta $?C$, tal que la longitud de cálculo de la $\mathbb{P}_r$-respuesta calculada admisible asociada $\langle \mathcal{A}; \sigma \rangle$ es $k < n$, entonces $(\dot{\mathcal{A}}; \sigma)$ es una $\diamond$-respuesta correcta difusa para $?C$. A continuación, lo probaremos para $k + 1$. Para ello, consideraremos que la longitud de cálculo de la $\mathbb{P}_r$-respuesta calculada admisible $\langle \mathcal{A}; \sigma \rangle$ es $k + 1$. Veremos que $(\dot{\mathcal{A}}; \sigma)$ es una $\diamond$-respuesta correcta difusa para $?A$, es decir, dado un $\diamond$-modelo arbitrario I para $\mathbb{P}$, necesitamos probar que se satisface

$$\dot{\mathcal{A}} \preceq I(A\sigma)$$

Dado que $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ es una $\mathbb{P}_r$-respuesta calculada difusa para $?A$, entonces el átomo A unifica con la cabeza $H_1\theta\theta_1$ de una regla en $\mathbb{P}_r$, que proviene de un conjunto de reglas críticas en $\mathbb{P}$, $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i \rangle \mid 1 \leq i \leq r\}$, con $\theta = \text{mgu}\{H_1, \ldots, H_r\}$. Por lo tanto, la regla en $\mathbb{P}_r$ es

$$\begin{aligned}\Big\langle H_1\theta\theta_1 \leftarrow @\big(&@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi^1_1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi^1_{n_1}), \ldots, \\ &@_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi^r_1, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi^r_{n_r})\big), \top\Big\rangle\end{aligned}$$

con $\theta_1 \in \mathcal{S}_g(H_1\theta)$, $\xi^i_1, \ldots, \xi^i_{n_i} \in \mathcal{S}_g(\mathcal{B}_i\theta\theta_1)$, para todo $i \in \{1, \ldots, r\}$, y existe una sustitución ν tal que $A\nu = H_1\theta\theta_1$. Además, el cuerpo de la regla puede ser escrito como

$$@'\left[B_1\theta\theta_1\eta^1_1, \ldots, B_1\theta\theta_1\eta^1_{m_1}, \ldots, B_p\theta\theta_1\eta^p_1, \ldots, B_p\theta\theta_1\eta^p_{m_p}\right]$$

donde $B_1, \ldots, B_p$ son los átomos que aparecen en los cuerpos de las reglas en $C_M^{H_1\theta}$, $@'$ es la composición de los símbolos de operadores que aparecen en el cuerpo del $\diamond$-reductante, y $\eta_1^1, \ldots, \eta_{m_p}^p \in \{\xi_1^1, \ldots, \xi_{n_r}^r\}$. Asimismo, el hecho de que $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ sea una $\mathbb{P}_r$-respuesta calculada difusa para $?A$ implica que existen

$$\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p \in L$$

tal que

$$\dot{\mathcal{A}} = \dot{@}' \left[\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p \right]$$

donde $\left\langle \dot{\mathcal{A}}_j^i; id \right\rangle$ es una $\mathbb{P}_r$-respuesta calculada difusa para $?B_i\theta\theta_1\eta_j^i$, que se obtiene de la secuencia

$$\left\langle B_i\theta\theta_1\eta_j^i; id \right\rangle \rightsquigarrow_{\text{AS}} \ldots \overset{(1)}{\rightsquigarrow}_{\text{AS}} \left\langle \mathcal{A}_j^i; \nu_j^i \left[\mathcal{V}ar(B_i\theta\theta_1\eta_j^i) \right] \right\rangle \overset{(2)}{\rightsquigarrow}_{\text{IS}} \left\langle \dot{\mathcal{A}}_j^i; id \right\rangle$$

para todo $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, m_i\}$. En (1), ν_j^i es el unificador más general entre $B_i\theta\theta_1\eta_j^i$ y la cabeza de una regla en $\mathbb{P}_r$, o es $\perp$ en el caso de que $B_i\theta\theta_1\eta_j^i$ no unifique con la cabeza de una regla en $\mathbb{P}_r$; mientras que (2) se verifica porque las cabezas de las reglas en $\mathbb{P}_r$ son básicas y, en consecuencia, las sustituciones $\nu_i^j = id$, para todo $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, m_i\}$. Por consiguiente, la $\mathbb{P}_r$-respuesta calculada difusa $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ se obtiene de la secuencia

$$\begin{array}{rl}
\langle A; id \rangle & \rightsquigarrow_{\text{AS}} \left\langle \top \& @' \left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p \right] \nu; \nu \right\rangle \\
& \overset{(*)}{=} \left\langle \top \& @' \left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p \right]; \nu \right\rangle \\
& \rightsquigarrow_{\text{AS}} \\
& \vdots \\
& \rightsquigarrow_{\text{AS}} \left\langle @' \left[\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p \right]; \nu \left[\mathcal{V}ar(A) \right] \right\rangle \\
& \rightsquigarrow_{\text{IS}} \left\langle \dot{@}' \left[\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p \right]; \nu \right\rangle
\end{array}$$

donde en (*) se ha tenido en cuenta que el cuerpo de la regla en $\mathbb{P}_r$ no tiene variables. Asimismo, como la longitud de cálculo de $\langle \mathcal{A}; \sigma \rangle$ es $k < n$, entonces cada $\mathbb{P}_r$-respuesta admisible calculada $\left\langle \mathcal{A}_j^i; id \right\rangle$ tiene una longitud

de cálculo menor que n. Por lo tanto, por la hipótesis de inducción, $(\mathcal{A}_j^i; id)$ es una $\diamond$-respuesta correcta difusa para $?B_i\theta\theta_1\eta_j^i$, y entonces, para el $\diamond$-modelo I dado, se obtiene la siguiente desigualdad

$$\dot{\mathcal{A}}_j^i \preceq I(B_i\theta\theta_1\eta_j^i) \tag{2.5}$$

para todo $i \in \{1,\ldots,p\}, j \in \{1,\ldots,m_i\}$. Finalmente, solo queda probar que $\dot{\mathcal{A}} \preceq I(A\nu)$, que resulta de la siguiente cadena de desigualdades.

$$\begin{aligned}
\dot{\mathcal{A}} &= \dot{@}'\left[\dot{\mathcal{A}}_1^1,\ldots,\dot{\mathcal{A}}_{m_1}^1,\ldots,\dot{\mathcal{A}}_1^p,\ldots,\dot{\mathcal{A}}_{m_p}^p\right] \\
&\overset{(1)}{\preceq} \dot{@}'\left[I(B_1\theta\theta_1\eta_1^1),\ldots,I(B_1\theta\theta_1\eta_{m_1}^1),\ldots,\dot{I}(B_p\theta\theta_1\eta_1^p),\ldots,I(B_p\theta\theta_1\eta_{m_p}^p)\right] \\
&= \hat{I}\left(@'\left[B_1\theta\theta_1\eta_1^1,\ldots,B_1\theta\theta_1\eta_{m_1}^1,\ldots,B_p\theta\theta_1\eta_1^p,\ldots,B_p\theta\theta_1\eta_{m_p}^p\right]\right) \\
&= \hat{I}\left(@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1,\ldots,\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1),\ldots,\right. \\
&\qquad \left.@_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r,\ldots,\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\right) \\
&= \dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^1),\ldots,\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_{n_1}^1)),\ldots, \\
&\qquad \dot{@}_{\sup}(\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_1^r),\ldots,\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r))) \\
&\overset{(2)}{=} \sup\left\{\dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^1),\ldots,\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_{n_1}^1)),\ldots,\right. \\
&\qquad \left.\dot{@}_{\sup}(\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_1^r),\ldots,\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r)))\xi \mid \xi \in \mathcal{S}_g\right\} \\
&\overset{(3)}{=} T_{\mathbb{P}_r}(I)(H_1\theta\theta_1) \\
&\overset{(4)}{\preceq} I(H_1\theta\theta_1) \\
&\preceq I(A\nu)
\end{aligned}$$

donde (1) se verifica por la monotonía de $\dot{@}'$ y la hipótesis de inducción (Expresión (2.5)); (2) del hecho de que solo hay una regla con cabeza $H_1\theta\theta_1$ en $\mathbb{P}_r$ y no hay variables en su cuerpo; (3) es consecuencia de la definición de $T_{\mathbb{P}_r}(I)(H_1\theta\theta_1)$; y (4) se cumple por el hecho de que I es un $\diamond$-modelo de $\mathbb{P}$ y, por la [128, Proposición 6], I es un modelo de $\mathbb{P}_r$. Por lo tanto, podemos concluir que $(\dot{\mathcal{A}}; \nu)$ es una $\diamond$-respuesta correcta difusa para $?A$ con respecto a $\mathbb{P}$. □

Una vez que la corrección del nuevo procedimiento ha sido probada, nos enfocamos en la completitud, para poder asegurar que todas las $\diamond$-respuestas correctas pueden calcularse o aproximarse a partir de $\mathbb{P}_r$-respuestas calculadas. Para ello, nos apoyamos sobre algunos resultados previos como [128, Proposición 8], en el que se prueba que cualquier iteración del operador de consecuencias inmediatas arbitrario $T_{\mathbb{P}}^{\diamond}$ es una $\mathbb{P}_r$-respuesta calculada. Posteriormente, presentamos los resultados de cuasicompletitud [128, Teoremas 8 y 9], donde el requisito de la continuidad del operador es fundamental. Específicamente se presentan varios resultados que permiten extender los teoremas de cuasicompletitud del caso clásico al marco de primer orden. El primer resultado de cuasicompletitud demuestra que toda $\diamond$-respuesta correcta difusa puede ser aproximada mediante una $\mathbb{P}_r$-respuesta calculada difusa. Por otro lado, el segundo resultado garantiza que para cada $\diamond$-respuesta correcta, existe una secuencia de elementos cuyo supremo supera dicha $\diamond$-respuesta correcta y, además, para cualquier elemento de esta secuencia, siempre habrá una $\mathbb{P}_r$-respuesta calculada que sea superior a dicho elemento.

En conclusión, hemos presentado una semántica operacional robusta y adecuada para cualquier aproximación de la programación lógica multiadjunta, basada en un operador de consecuencias inmediatas monótono y arbitrario, en el contexto de lógica de primer orden. Sin embargo, aún queda por analizar la $\mathbb{P}_r$-respuesta calculada máxima, con el fin de poder establecer su correspondencia con la θ-$\diamond$-respuesta correcta máxima.

Capítulo 3

Operador de consecuencias inmediatas con cuantificadores generalizados

Título:	Immediate consequences operator on generalized quantifiers
Autores:	Jesús Medina, José Antonio Torné-Zambrano
Revista:	Fuzzy Sets and Systems
Factor de impacto JCR:	3,9. Posición 12 de 267 (Q1)
Año:	2022
Categoría:	Mathematics, Applied
Publicación:	19 de agosto de 2022
DOI:	https://doi.org/10.1016/j.fss.2022.08.014

Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems 456 (2023) 72–91

FUZZY sets and systems

www.elsevier.com/locate/fss

Immediate consequences operator on generalized quantifiers ☆

Jesús Medina, José Antonio Torné-Zambrano *

Department of Mathematics, University of Cádiz, Cádiz, Spain

Received 6 July 2021; received in revised form 29 July 2022; accepted 9 August 2022
Available online 19 August 2022

Abstract

The semantics of a multi-adjoint logic program is usually defined through the immediate consequences operator $T_{\mathbb{P}}$. However, the definition of the immediate consequences operator as the supremum of a set of values can provide some problem when imprecise datasets are considered, due to the strict feature of the supremum operator. Hence, based on the flexibility of generalized quantifiers to weaken the existential feature of the supremum operator, this paper presents a generalization of the immediate consequences operator with interesting properties for solving the aforementioned problem.
© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Keywords: Fuzzy sets; Logic programming; Immediate consequences operator; Generalized quantifiers

1. Introduction

Logic is the field of mathematics that is responsible for studying valid schemes of human reasoning. Since the advent of computers, one of the greatest applications of logic has been in the field of computing as a programming language and as a model of computing [19]. From the studies of different authors in these research areas, logic programming arose as a revolutionary idea for using logic as a programming language.

Semantics of a logic program has been characterized through the post-fixed points of the immediate consequences operator $T_{\mathbb{P}}$ [16,20–22,26–28]. Specifically, it is proved that the meaning of a logic program $\mathbb{P}$ is obtained as the least fixed point of the operator $T_{\mathbb{P}}$. By the definition of this operator as the supremum of a set of values in diverse fuzzy frameworks [7,16,21,22,26], the presence of some (small) errors in the data (noise) can disrupt the final result. Consequently, the main goal of this paper is to provide a further generalization of the framework given in [21] so that it is possible to solve this problem. To proceed in this way, we will use generalized quantifiers [3,4,10,25] in order to relieve the possible presence of this noise in the data. These quantifiers are particular cases of fuzzy quantifiers [1,2,14,15], which are mathematical tools with the ability to model a diverse class of natural language such as “Many”,

☆ Partially supported by the 2014-2020 ERDF Operational Programme in collaboration with the State Research Agency (AEI) in project PID2019-108991GB-I00, and with the Department of Economy, Knowledge, Business and University of the Regional Government of Andalusia in project FEDER-UCA18-108612, and by the European Cooperation in Science & Technology (COST) Action CA17124.
* Corresponding author.
E-mail addresses: jesus.medina@uca.es (J. Medina), joseantonio.torne@uca.es (J.A. Torné-Zambrano).

https://doi.org/10.1016/j.fss.2022.08.014
0165-0114/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

"At least", "A few" or "Most". Moreover, generalized quantifiers are very versatile and can be adapted to a big range of situations [8,12,24].

Taking into account that standard immediate consequences operator implicitly employs the existential quantifier in fuzzy logic programming, it can be generalized using other quantifiers determined by other fuzzy measures that can be useful in dealing with other sort of situations in order to create a more flexible and softer framework, where the possible noise presented in the data does not affect so much the final computations.

Thus, this paper will use generalized quantifiers weaker than the supremum in the immediate consequences operator and so, decrease the impact of imprecise data (noise) in computations. Specifically, two alternatives will be analyzed and the main properties will be studied. Since generalized quantifiers range from the existential to the universal quantifiers, an orness measure will be considered to be sure that the considered measure provides quantifiers closer to the existential quantifiers (as the supremum is) than the universal one.

The organization of the paper is as follows: preliminary notions and results of fixed points theory, generalized quantifiers, multi-adjoint structures, and multi-adjoint logic programming are introduced in Section 2; Section 3 presents the first approach of the immediate consequences operator using generalized quantifiers. With the objective of evaluating the different generalizations of the immediate consequences operator, we introduce a generalization of the concept of orness measure in Section 4. Finally, Section 5 introduces several conclusions and pointers for further research.

2. Preliminaries

This section recalls some important results which will be useful in the following sections of the paper. For more details see [21,22].

2.1. Multi-adjoint algebra structure

In the initial work on multi-adjoint logic programming [21], adjoint pairs were considered as basic operators.

Definition 1. (*Adjoint pair*). Let $\langle P, \preceq\rangle$ be a poset and $(\leftarrow, \&)$ a pair of binary operators in P such that

1. The operator $\&$ is increasing in both arguments.
2. The operator $\leftarrow$ is increasing in the first argument (consequent) and decreasing in the second one (antecedent).
3. The adjoint property holds, that is, $x \preceq (y \leftarrow z)$ if and only if $(x \& z) \preceq y$, for all $x, y, z \in P$.

Under these conditions, we say that $(\leftarrow, \&)$ is an *adjoint pair* in $\langle P, \preceq\rangle$.

These operators are used in the fuzzy logic programming framework of this paper on complete lattices.

Definition 2. (*Multi-adjoint lattice*). Let $\langle L, \preceq\rangle$ be a complete lattice. A *multi-adjoint lattice* $\mathcal{L}$ is a tuple $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ satisfying

1. $\langle L, \preceq\rangle$ is bounded, i.e., it has a bottom $\bot$ and a top $\top$ elements.
2. $(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq\rangle$, for all $i \in \{1, \ldots, n\}$.
3. $\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$, for all $\vartheta \in L$ and $i \in \{1, \ldots, n\}$.

Moreover, other general increasing operators can be considered in multi-adjoint logic programming, such as aggregator operators.

Definition 3. (*Aggregation function*). A mapping $@ : [0, 1]^n \to [0, 1]$ is said to be an *aggregation function* if:

1. $@$ satisfies the boundary conditions $@(0, \ldots, 0) = 0$ and $@(1, \ldots, 1) = 1$.
2. $@$ is increasing, that is, for each $i \in \{1, \ldots, n\}$, if $x_i \preceq y$, then $@(x_1, \ldots, x_n) \preceq @(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$, for all $y, x_1, \ldots, x_n \in [0, 1]$.

2.2. Multi-adjoint logic programming

The algebraic structure considered in this framework is recalled next. The notions of Ω-algebra, graded set, alphabet, and other basic definitions can be seen in [20,21].

Definition 4. (*Multi-adjoint Ω-algebra*). Let Ω be a graded set containing operators $\leftarrow_i$ and $\&_i$, for $i \in \{1, \ldots, n\}$, and possibly some extra operators, and let $\mathfrak{L} = \langle L, I\rangle$ be an Ω-algebra whose carrier set L is a complete lattice under $\preceq$. We say that $\mathfrak{L}$ is a *multi-adjoint Ω-algebra* with respect to the pairs $(\leftarrow_i, \&_i)$, for $i \in \{1, \ldots, n\}$, if $\mathcal{L} = (L, \preceq, I(\leftarrow_1), I(\&_1), \ldots, I(\leftarrow_n), I(\&_n))$ is a multi-adjoint lattice. $I(\leftarrow_i)$ can also be denoted as $\dot{\leftarrow}_i$ and $I(\&_i)$ as $\dot{\&}_i$.

Example 1. $\langle [0, 1], I\rangle$ is a multi-adjoint Ω-algebra with respect to the graded set $\Omega = \{\leftarrow_{\mathrm{P}}, \&_{\mathrm{P}}, \leftarrow_{\mathrm{G}}, \&_{\mathrm{G}}, \wedge_{Ł}, @\}$, where $\&_{\mathrm{P}}$ and $\&_{\mathrm{G}}$ are the product and Gödel t-norms respectively, together with their residuated implications, $\wedge_{Ł}$ is the Łukasiewicz t-norm and @ is an aggregation operator. These operators are defined for all $x, y, z \in [0, 1]$ as follows

$$\&_{\mathrm{G}}(x, y) = \min\{x, y\} \qquad z \leftarrow_{\mathrm{G}} y = \begin{cases} 1 & \text{if} \quad y \leq z \\ z & \text{otherwise} \end{cases}$$

$$\&_{\mathrm{P}}(x, y) = x \cdot y \qquad z \leftarrow_{\mathrm{P}} y = \begin{cases} 1 & \text{if} \quad y \leq z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$

$$\wedge_{Ł}(x, y) = \max\{0, x + y - 1\} \qquad @(x, y) = \frac{2x + y}{3}$$

□

2.2.1. Syntax of multi-adjoint logic programming

Multi-adjoint logic programs will be constructed from the abstract syntax induced by a multi-adjoint algebra on a set of propositional symbols. Specifically, we will consider a multi-adjoint Ω-algebra $\mathfrak{L}$ whose extra operators are either conjunctions, denoted by $\wedge_1, \ldots, \wedge_k$, or disjunctions, denoted by $\vee_1, \ldots, \vee_l$ or aggregators, denoted by $@_1, \ldots, @_m$. Furthermore, let Π be a set of propositional symbols and the corresponding algebra of formulas $\mathfrak{F}$ freely generated from Π by the operators in Ω [21]. This algebra will be used to define the syntax of a propositional language.

Definition 5. (*Multi-adjoint logic program*). A *multi-adjoint logic program* is a set $\mathbb{P}$ of weighted rules of the form $\langle (A \leftarrow_i \mathcal{B}), \vartheta\rangle$, also denoted $A \overset{\vartheta}{\leftarrow} \mathcal{B}$, such that:

1. $(A \leftarrow_i \mathcal{B})$ is a formula of $\mathfrak{F}$, which is simply called *rule (of the program)*.
2. The confidence factor ϑ is an element (a truth-value) of L.
3. The head of the rule A is a propositional symbol of Π.
4. The body formula $\mathcal{B}$ is a formula of $\mathfrak{F}$ built from propositional symbols $B_1, \ldots, B_n$ ($n \geq 0$) by the use of conjunctions $\&_1, \ldots, \&_n$ and $\wedge_1, \ldots, \wedge_k$, disjunctions $\vee_1, \ldots, \vee_l$ and aggregators $@_1, \ldots, @_m$.

The particular case of a weighted rule where the body $\mathcal{B}$ is $\top$ is called a *fact*. Notice that the top element of the lattice is considered as a constant in the Ω-graded set used for computing the well-formed formulas set F_Ω.

Sometimes, we will represent a rule $A \leftarrow_i \mathcal{B}$ as $A \leftarrow @[B_1, \ldots, B_n]$, where @ is the aggregator obtained as the composition of all operational symbols in $\mathcal{B}$, and $B_1, \ldots, B_n$ are propositional symbols which appear in $\mathcal{B}$ and so, $\mathcal{B} = @[B_1, \ldots, B_n]$.

Example 2. In Example 1, identifying the operator symbols in Ω with their interpretations, we can consider the multi-adjoint Ω-algebra

$$\langle [0, 1], \leq, \leftarrow_{\mathrm{G}}, \&_{\mathrm{G}}, \leftarrow_{\mathrm{P}}, \&_{\mathrm{P}}, \wedge_{Ł}, @\rangle$$

where $\&_G$ and $\&_P$ are the Gödel and the product conjunctions respectively, with $\leftarrow_G$ and $\leftarrow_P$ being their respective adjoint implications. Moreover, we use the extra operator $\wedge_Ł$, the Łukasiewicz conjunction and the aggregator @ given in Example 1. We define the set of propositional symbols Π as

$$\Pi = \{\texttt{covid}, \texttt{diarrhoea}, \texttt{headache}, \texttt{moderate fever}, \texttt{cough}, \texttt{waist pain}\}$$

We can consider, in an specific situation, the multi-adjoint logic program $\mathbb{P}_1$ defined through the rules

$$\texttt{covid} \overset{0.8}{\leftarrow_G} @\,(\texttt{diarrhoea} \wedge_Ł \texttt{headache}, \texttt{waist pain})$$
$$\texttt{covid} \overset{0.9}{\leftarrow_P} \texttt{moderate fever}$$
$$\texttt{covid} \overset{0.7}{\leftarrow_P} \texttt{cough}$$

These rules are completed with observations about each particular person, in the form of *facts*. For example, a particular individual may have the symptoms represented by the following facts.

$$\begin{aligned}
\texttt{moderate fever} &\overset{0.6}{\leftarrow_P} 1\\
\texttt{cough} &\overset{0.8}{\leftarrow_P} 1\\
\texttt{diarrhoea} &\overset{0.6}{\leftarrow_P} 1\\
\texttt{headache} &\overset{0.7}{\leftarrow_P} 1\\
\texttt{waist pain} &\overset{0.3}{\leftarrow_P} 1
\end{aligned}$$

□

2.2.2. Semantics of multi-adjoint logic programming

Semantics of a multi-adjoint logic program allows us to give meaning to formulas through its syntactic structure. Let Ω be a graded set, Π a set of propositional symbols, $\mathfrak{F}$ the Ω-algebra of well-formed formulas with carrier F_Ω and $\mathfrak{L}$ an arbitrary multi-adjoint Ω-algebra with carrier L.

Definition 6. (*Interpretation*). An *interpretation* is a mapping $I : \Pi \longrightarrow L$ which assigns values of the complete lattice L to the set of propositional symbols. The set of all interpretations is denoted as $\mathcal{I}_\mathfrak{L}$.

The set of interpretations $\mathcal{I}_\mathfrak{L}$ together with the point-wise ordering forms a complete lattice [21], denoted by $\langle \mathcal{I}_\mathfrak{L}, \sqsubseteq \rangle$. The least interpretation $\triangle$ maps every propositional symbol to the least element $\bot$ of L.

From the unique homomorphic extension [21], each interpretation uniquely extends to the set of formulas F_Ω, i.e., $\hat{I} : F_\Omega \longrightarrow L$. The ordering $\leq$ defined in the set of truth-values L can be extended to the set of interpretations.

Definition 7. (*Satisfaction and model*). Given an interpretation $I \in \mathcal{I}_\mathfrak{L}$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$ is *satisfied* by I if and only if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$. An interpretation $I \in \mathcal{I}_\mathfrak{L}$ is a *model* of a multi-adjoint logic program $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.

It is possible to generalize the immediate consequences operator, given by van Emden and Kowalski [13], to the framework of multi-adjoint logic programming as follows:

Definition 8. (*Immediate consequences operator*). Let $\mathbb{P}$ be a multi-adjoint logic program. The *immediate consequences operator* $T^\mathfrak{L}_\mathbb{P} : \mathcal{I}_\mathfrak{L} \longrightarrow \mathcal{I}_\mathfrak{L}$, mapping interpretations to interpretations, is defined by considering

$$T^\mathfrak{L}_\mathbb{P}(I)(A) = \sup\left\{\vartheta \,\dot{\&}_i\, \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow_i} \mathcal{B} \in \mathbb{P}\right\}$$

From now on, we will simply refer to it as $T_\mathbb{P}$. Notice that it makes sense to consider the suprema due to the fact that L is assumed to be a complete lattice. Next, we are going to characterize the semantics of a multi-adjoint logic program as the post-fixed points of the operator $T_\mathbb{P}$.

Theorem 1 *([21]). An interpretation $I \in \mathcal{I}_\mathfrak{L}$ is a model of a multi-adjoint logic program $\mathbb{P}$ if and only if it is a post-fixed point of $T_\mathbb{P}$, that is $T_\mathbb{P}(I) \sqsubseteq I$.*

In addition, the following result for the immediate consequences operator holds.

Theorem 2 *([21]). The operator $T_{\mathbb{P}}$ is monotonic.*

From Theorems 1 and 2, if I is a model, then $T_{\mathbb{P}}(I)$ is a model. Once we have proved the monotonicity of $T_{\mathbb{P}}$, the following result shows that it has a least fixed point.

Proposition 1 *([27]). The operator $T_{\mathbb{P}} : \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ has a least fixed point $lfp(T_{\mathbb{P}})$ verifying that*

$$lfp(T_{\mathbb{P}}) = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}(I) \sqsubseteq I\}$$

Next, we are going to relate the least fixed point of $T_{\mathbb{P}}$ with a particular model. From Theorem 1, we can characterize the least model $M_{\mathbb{P}}$ as the least fixed point of the operator $T_{\mathbb{P}}$, allowing us to define the semantics of $\mathbb{P}$ in this way. In addition, the least model can be obtained by iterating an ordinal number of times α the operator $T_{\mathbb{P}}$, starting from the least interpretation $\triangle$.

Theorem 3 *([21]). Let L be a complete lattice and $\mathbb{P}$ a multi-adjoint logic program. Then, for some ordinal α it is satisfied that $M_{\mathbb{P}} = lfp\,(T_{\mathbb{P}}) = T^{\alpha}_{\mathbb{P}}(\triangle)$.*

Example 3. Taking into account Example 2, we are going to show how to calculate the value of the immediate consequences operator and the least model of the program $\mathbb{P}_1$. From Theorem 3, the least model of $\mathbb{P}_1$ is obtained through iterating the $T_{\mathbb{P}_1}$ starting from the least interpretation $\triangle$. From the unique homomorphic extension, we can obtain

$$\begin{aligned} T_{\mathbb{P}_1}(\triangle)(\texttt{covid}) &= \sup\big\{0.9\dot{\&}_{\mathrm{P}}\widehat{\triangle}(\texttt{moderate fever}), 0.7\dot{\&}_{\mathrm{P}}\widehat{\triangle}(\texttt{cough}),\\ &\qquad 0.8\dot{\&}_{\mathrm{G}}@\left(\widehat{\triangle}(\texttt{diarrhoea} \wedge_{Ł} \texttt{headache}), \widehat{\triangle}\,(\texttt{waist pain})\right)\big\}\\ &= \sup\{0, 0, 0\}\\ &= 0 \end{aligned}$$

$$\begin{aligned} T^2_{\mathbb{P}_1}(\triangle)(\texttt{covid}) &= \sup\big\{0.9\dot{\&}_{\mathrm{P}}\widehat{T_{\mathbb{P}_1}}(\triangle)(\texttt{moderate fever}), 0.7\dot{\&}_{\mathrm{P}}\widehat{T_{\mathbb{P}_1}}(\triangle)(\texttt{cough}),\\ &\qquad 0.8\dot{\&}_{\mathrm{G}}@\left(\widehat{T_{\mathbb{P}_1}}(\triangle)(\texttt{diarrhoea} \wedge_{Ł} \texttt{headache}), \widehat{T_{\mathbb{P}_1}}(\triangle)\,(\texttt{waist pain})\right)\big\}\\ &= \sup\{0.54, 0.56, 0.3\}\\ &= 0.56 \end{aligned}$$

In the following table we show the iteration of the $T_{\mathbb{P}_1}$. Remark that the least fixed point is reached in the second iteration; hence Theorem 3 holds that the least model of $\mathbb{P}_1$ corresponds to $T^2_{\mathbb{P}_1}(\triangle)$.

	$\triangle$	$T_{\mathbb{P}_1}(\triangle)$	$T^2_{\mathbb{P}_1}(\triangle)$	$T^3_{\mathbb{P}_1}(\triangle)$
`moderate fever`	0	0.6	0.6	0.6
`cough`	0	0.8	0.8	0.8
`diarrhoea`	0	0.6	0.6	0.6
`headache`	0	0.7	0.7	0.7
`waist pain`	0	0.3	0.3	0.3
`covid`	0	0	0.56	0.56

Notice that any interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ greater than the least fixed point of $T_{\mathbb{P}}$, that is $\mathrm{lfp}(T_{\mathbb{P}}) \sqsubseteq I$, does not need to be a model of the program $\mathbb{P}$. We will present a counterexample. For this we consider Example 2 and the interpretation $I : \Pi \longrightarrow [0, 1]$ defined as $I(\texttt{covid}) = 0.6$ and 0.9 for the rest.

As $I(\texttt{covid}) = 0.6 \leq 0.81 = T_{\mathbb{P}_1}(I)(\texttt{covid})$ then it is not verified that $T_{\mathbb{P}_1}(I) \sqsubseteq I$, indeed, $T_{\mathbb{P}_1}(I)$ and I are incomparable. Consequently, we have found an interpretation I such that $\mathrm{lfp}(T_{\mathbb{P}}) \sqsubseteq I$, but I is not a model of $\mathbb{P}$. □

2.3. Generalized quantifiers

In order to understand the generalization proposed in this paper, it is inevitable to introduce some basic aspects related to generalized quantifier [3,4,10,25]. First of all, we need to present the definitions of fuzzy measure and the invariance of fuzzy measure with respect to the cardinality.

Definition 9 *([18]).* (*Fuzzy measure*). Given a non-empty finite set U and a σ-algebra $\mathscr{A}$ of subsets of U, we say that $\mu\colon \mathscr{A} \longrightarrow [0,\infty)$ is a *fuzzy measure* if it holds the following two conditions:

1. $\mu(\varnothing) = 0$.
2. For all $A, B \in \mathscr{A}$ such that $A \subseteq B$, $\mu(A) \leq \mu(B)$.

The general definition of fuzzy measures involves a fourth axiom (about continuity) which does not apply in the framework of this paper (since we consider a finite and discrete set U). Moreover, although the definition given in [18] contemplates $\mu(U) = 1$, this property is considered as a separated one, as it is used in [3]. We say that μ is a *normalized fuzzy measure* if $\mu(U) = 1$. From now on, we will call it simply a fuzzy measure.

Definition 10 *([18]).* (*Invariance with respect to the cardinality*). Let U be a non-empty finite universe, $\mathcal{P}(U)$ the powerset of U and $\mu\colon \mathcal{P}(U) \longrightarrow [0,1]$ a fuzzy measure. We will say that μ is *invariant with respect to the cardinality*, if the following condition holds

$$\text{If } |A| = |B|, \text{ then } \mu(A) = \mu(B), \text{ for all } A, B \in \mathcal{P}(U)$$

where $|\cdot|$ denotes the cardinality of a set.

From this definition, we can introduce the notion of generalized quantifier.

Definition 11 *([3,25]).* (*Generalized quantifier determined by a fuzzy measure μ*). Let U be a non-empty finite universe, $\mathcal{P}(U)$ the powerset of U, $\mu\colon \mathcal{P}(U) \to [0,1]$ a fuzzy measure invariant with respect to the cardinality and $\otimes$ an operator which is increasing on the first argument and satisfies the following boundary conditions: $x \otimes 1 = x$, $x \otimes 0 = 0 \otimes x = 0$, for all $x \in [0,1]$. A mapping $Q_\mu\colon \mathcal{F}(U) \to [0,1]$ defined, for all $C \in \mathcal{F}(U) = \{C \mid C\colon U \to [0,1]\}$, as

$$Q_\mu(C) = \bigvee_{D \in \mathcal{P}(U)\setminus\{\varnothing\}} \left(\left(\bigwedge_{u \in D} C(u) \right) \otimes \mu(D) \right)$$

is called *generalized quantifier determined by the fuzzy measure μ*, where $\bigvee$ and $\bigwedge$ represent the supremum and infimum operators in the unit interval.

Note that in Definition 11 we have not considered the empty set as in the definition of generalized quantifiers given in [3,25]. Indeed, we cannot consider it because the operator $\otimes$ does not need to satisfy the boundary conditions with the bottom element due to, although it satisfies the boundary conditions with the top element in the right side, it could not be increasing on the right side. It is clear that if $\otimes$ is left-continuous on the right side then the empty set could be considered and the definition be simplified as in [6].

Furthermore, notice that this definition of generalized quantifiers is a particular case of the Sugeno integral considering $\wedge$ as the operator $\otimes$. The definition of the Sugeno-like integral on L, where L is a complete residuated lattice, was provided independently in [9,11].

Next, we present two important examples of generalized quantifiers, which remark that this definition is a proper generalization of the existential and universal quantifiers, and also show the great versatility of the definition of generalized quantifier and its dependence on the considered fuzzy measure.

Example 4. The universal and existential quantifiers can be obtained from Definition 11 considering the minimum and maximum fuzzy measures $\mu^\forall$ and $\mu^\exists$ respectively, which are defined as follows:

$$\mu^{\forall}(D) = \begin{cases} 1 & \text{if} \quad D = U \\ 0 & \text{if} \quad D \neq U \end{cases} \qquad \mu^{\exists}(D) = \begin{cases} 0 & \text{if} \quad D = \varnothing \\ 1 & \text{if} \quad D \neq \varnothing \end{cases}$$

Furthermore, it can be proved that the following equations are satisfied [5]:

$$Q^{\forall}(C) = \bigwedge_{u \in U} C(u) \qquad Q^{\exists}(C) = \bigvee_{u \in U} C(u) \qquad \square$$

The following result will provide a characterization of generalized quantifiers that will be very useful for the generalization of the immediate consequences operator, which will be studied in the next sections.

Theorem 4 *([25]). Let Q_μ be a generalized quantifier in the non-empty finite universe $U = \{u_1, \ldots, u_n\}$ determined by a fuzzy measure μ invariant with respect to the cardinality. Then, for all $C \in \mathcal{F}(U)$,*

$$Q_\mu(C) = \bigvee_{i=1}^{n} C\left(u_{\pi(i)}\right) \otimes \mu\left(\{u_1, \ldots, u_i\}\right)$$

where π is a permutation on $\{1, 2, \ldots, n-1, n\}$ such that

$$C\left(u_{\pi(n)}\right) \leq C\left(u_{\pi(n-1)}\right) \leq \ldots \leq C\left(u_{\pi(2)}\right) \leq C\left(u_{\pi(1)}\right)$$

3. Immediate consequences operator generalized through generalized quantifiers

In this section we are going to generalize the immediate consequences operator with the use of generalized quantifiers. First of all, we need to introduce the problem related to the definition of the $T_{\mathbb{P}}$ given through the supremum of a set of values.

Example 5. Consider Example 2, with the interpretation I defined as $I(\texttt{headache}) = 0.7$, $I(\texttt{moderate fever}) = 0.9$, $I(\texttt{diarrhoea}) = 0.8$, $I(\texttt{cough}) = 0.7$ and $I(\texttt{waist pain}) = 0.5$. Then, we obtain that

$$\begin{aligned} T_{\mathbb{P}_1}(I)(\texttt{covid}) = \sup \Big\{ & 0.9 \dot{\&}_{\mathrm{P}} \hat{I}(\texttt{moderate fever}), 0.7 \dot{\&}_{\mathrm{P}} \hat{I}(\texttt{cough}), \\ & 0.8 \dot{\&}_{\mathrm{G}} @ \left(\hat{I}(\texttt{diarrhoea} \wedge_{\mathrm{L}} \texttt{headache}), \hat{I}(\texttt{waist pain}) \right) \Big\} \\ = \sup & \{0.81, 0.49, 0.5\} \\ = & \, 0.81 \end{aligned}$$

With this example we can see that not all of the information is considered in the computation of the immediate consequences operator, being very sensitive to possible errors in the values of the observed variables. For instance, in this case, we have considered 0.81, which is considerably greater than the other two values. $\square$

The main goal of this section is to provide a more general definition of the immediate consequences operator which provides a possible solution to the problem of the existence of possible noise in the dataset. The first approach will be consider other data different from the supremum one. For that, we are going to use generalized quantifiers. From now on, we will use a multi-adjoint logic program $\mathbb{P}$ over a set of propositional symbols Π and a multi-adjoint Ω-algebra $\mathfrak{L}$ with [0, 1] as lattice and with respect to the adjoint pairs $(\leftarrow_i, \&_i)$, for all $i \in \{1, \ldots, n\}$.

3.1. Definition of the $T_{\mathbb{P}}$ through the existential quantifier

First of all, we introduce the definition of the mapping C_I^A, associated with a propositional symbol A and an interpretation I, which will facilitate the achievement of the stated goal.

Definition 12. (*Mapping C_I^A*). Let $\mathbb{P}$ be a multi-adjoint logic program and I an interpretation of $\mathbb{P}$. We define the mapping $C_I^A \colon \mathbb{P} \longrightarrow [0, 1]$, for each $\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}$, as follows

$$C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle\right) = \begin{cases} \vartheta \,\dot{\&}_i\, \hat{I}(\mathcal{B}) & \text{if } C = A \\ 0 & \text{otherwise} \end{cases}$$

From this definition, we can present the following result, in which we make a comparison that will allow us to generalize the immediate consequences operator with the use of generalized quantifiers.

Proposition 2. *Let $\mathbb{P}$ be a multi-adjoint logic program. Then, it is satisfied that*

$$Q^{\exists}\left(C_I^A\right) = T_{\mathbb{P}}(I)(A)$$

where $\mathbb{P}$ is the considered universe, $Q^{\exists}$ is the existential quantifier defined on $\mathbb{P}$ in Example 4, and C_I^A is the mapping given by Definition 12.

Proof. Considering the mapping C_I^A given by Definition 12, the definition of immediate consequences operator and the characterization of the existential quantifier given by Example 4, we have the following chain of equalities.

$$\begin{aligned} Q^{\exists}\left(C_I^A\right) &= \bigvee_{\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}} C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle\right) \\ &= \bigvee_{\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}} C_I^A\left(\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle\right) \\ &= \bigvee_{\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}} \left(\vartheta \,\dot{\&}_i\, \hat{I}(\mathcal{B})\right) \\ &= \sup\left\{\vartheta \,\dot{\&}_i\, \hat{I}(\mathcal{B}) \mid \langle A \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}\right\} \\ &= T_{\mathbb{P}}^{\mathfrak{L}}(I)(A) \end{aligned}$$

□

Notice that in the case of a propositional symbol A that is not the head of any rule, $T_{\mathbb{P}}(I)(A) = 0$ and $Q^{\exists}(C_I^A) = 0$, for any interpretation I.

Once we have established the characterization of $T_{\mathbb{P}}$ with the existential quantifier, in the next section we will use it to introduce a generalization of the immediate consequences operator in terms of generalized quantifiers.

3.2. Definition of the $T_{\mathbb{P}}$ through an arbitrary generalized quantifier

To generalize the immediate consequences operator through a generalized quantifier, we are going to use the same mapping C_I^A given in Definition 12, together with the definition of generalized quantifier determined by a fuzzy measure μ (Definition 11).

Definition 13. (*Quantified immediate consequences operator*). Let $\mathbb{P}$ be a non-empty multi-adjoint logic program, $\mu\colon \mathcal{P}(\mathbb{P}) \longrightarrow [0,1]$ a fuzzy measure invariant with respect to the cardinality and $\otimes$ an operator which is increasing in the first argument and satisfies the following boundary conditions: $x \otimes 1 = x$, $x \otimes 0 = 0 \otimes x = 0$, for all $x \in [0,1]$. We define the *quantified immediate consequences operator* obtained from the generalized quantifier Q_μ as the mapping $T_{\mathbb{P}}^{Q_\mu}\colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ given by

$$T_{\mathbb{P}}^{Q_\mu}(I)(A) = Q_\mu\left(C_I^A\right) = \bigvee_{D \in \mathcal{P}(\mathbb{P})} \left(\left(\bigwedge_{\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in D} C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle\right)\right) \otimes \mu(D)\right)$$

for all $I \in \mathcal{I}_{\mathfrak{L}}$ and $A \in \Pi$.

The section above introduced the trivial example in which the existential quantifier is considered, which actually coincides with the classical definition of $T_{\mathbb{P}}$. Next, we will take into account another example in which the generalized quantifier selects the second greatest element in the computation of $T_{\mathbb{P}}$ instead of the supremum.

Example 6. Consider a finite multi-adjoint logic program $\mathbb{P}$ with rules $\{u_1, \ldots, u_n\}$, an interpretation I, the mapping C_I^A given by Definition 12 and the fuzzy measure $\mu^1 \colon \mathcal{P}(\mathbb{P}) \longrightarrow [0,1]$ defined as

$$\mu^1(D) = \begin{cases} 0 & \text{if } D = \varnothing \text{ or } D = \{u_i\} \text{ for all } i \in \{1, \ldots, n\} \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

for all $D \in \mathcal{P}(\mathbb{P})$. Although we have already mentioned it, notice that the measure rejects the highest value. Taking into account that the fuzzy measure stops being null when the cardinality of the set is equal or greater than 2, the quantifier associated to this fuzzy measure will only take these sets into account. For this reason, the quantifier selects the second highest value.

We can order the elements $C_I^A(u_i)$ for each $i \in \{1, \ldots, n\}$ through the permutation π of the set of rules $\{u_1, \ldots, u_n\}$ satisfying that

$$C_I^A\left(u_{\pi(n)}\right) \leq C_I^A\left(u_{\pi(n-1)}\right) \leq \ldots \leq C_I^A\left(u_{\pi(2)}\right) \leq C_I^A\left(u_{\pi(1)}\right)$$

Taking into account this permutation, Definition 13, and the characterization of the generalized quantifiers given by Theorem 4, we have that

$$T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) = Q_{\mu^1}(C_I^A) = \bigvee_{i=1}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1, \ldots, u_i\}\right) = C_I^A\left(u_{\pi(2)}\right) \qquad \square$$

Now, we are going to prove the monotonicity of the operator $T_{\mathbb{P}}^{Q_\mu}$ through the following results. The first one shows that generalized quantifiers given in Definition 11 are increasing.

Proposition 3. *The quantifier Q_μ is increasing in $\mathcal{F}(U)$ for each fuzzy measure μ invariant with respect to the cardinality.*

Proof. Consider two mappings $C_1, C_2 \in \mathcal{F}(U) = \{C \mid C \colon U \longrightarrow [0,1]\}$ verifying that $C_1 \leq C_2$. We need to prove that $Q_\mu(C_1) \leq Q_\mu(C_2)$.

As $C_1 \leq C_2$ then, for each $u \in U$, it holds that $C_1(u) \leq C_2(u)$. Thus, for each $i \in \{1, \ldots, n\}$, due to the monotonicity of the operator $\otimes$ on the first argument, it is verified that

$$C_1(u_{\pi(i)}) \otimes \mu\left(\{u_1, \ldots, u_i\}\right) \leq C_2(u_{\pi(i)}) \otimes \mu\left(\{u_1, \ldots, u_i\}\right)$$

where π is a permutation in $\{1, 2, \ldots, n-1, n\}$ such that $C\left(u_{\pi(n)}\right) \leq C\left(u_{\pi(n-1)}\right) \leq \ldots \leq C\left(u_{\pi(2)}\right) \leq C\left(u_{\pi(1)}\right)$.

Taking suprema in the previous expression and taking into account the characterization of quantifiers given in Theorem 4 we can write

$$\begin{aligned} Q_\mu(C_1) &= \bigvee_{i=1}^{n} C_1(u_{\pi(i)}) \otimes \mu\left(\{u_1, \ldots, u_i\}\right) \\ &\leq \bigvee_{i=1}^{n} C_2(u_{\pi(i)}) \otimes \mu\left(\{u_1, \ldots, u_i\}\right) \\ &= Q_\mu(C_2) \end{aligned}$$

We conclude that the operator Q_μ is increasing for each μ. $\square$

The following result proves that the mapping defined in Definition 12 is increasing.

Proposition 4. *The mapping C_I^A is increasing in the lattice of interpretations $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$.*

Proof. Consider the interpretations $I_1, I_2 \in \mathcal{I}_{\mathfrak{L}}$, such that it is verified that $I_1 \sqsubseteq I_2$, i.e., $I_1(p) \leq I_2(p)$, for all $p \in \Pi$. Let $\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle$ be a rule of the program $\mathbb{P}$; we have to prove that $C_{I_1}^A(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle) \leq C_{I_2}^A(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle)$.

If $C \neq A$, then it is obvious because in both cases it would be 0.

If $C = A$, if we have that $\hat{I}_1(\mathcal{B}) \leq \hat{I}_2(\mathcal{B})$, for the monotonicity of the operators $\dot{\&}_i$, it holds that $\vartheta \dot{\&}_i \hat{I}_1(\mathcal{B}) \leq \vartheta \dot{\&}_i \hat{I}_2(\mathcal{B})$ and we could conclude that C is increasing in $\mathcal{I}_{\mathfrak{L}}$.

Therefore, we need to prove that $\hat{I}_1(\mathcal{B}) \leq \hat{I}_2(\mathcal{B})$ for all $\mathcal{B}$. First of all, assume that $\mathcal{B}$ is an atom. From the homomorphic extension theorem, it is verified that $\hat{I}_1(\mathcal{B}) = I_1(\mathcal{B}) \leq I_2(\mathcal{B}) = \hat{I}_2(\mathcal{B})$.

Consider the body $\mathcal{B} = @[B_1, \ldots, B_n]$, where $@$ is the aggregator obtained as the composition of all operational symbols in $\mathcal{B}$ and $B_1, \ldots, B_n$ are propositional symbols which appear in $\mathcal{B}$ (see Page 74).

By the comment above, since each B_i is an atom, we have that $I_1(B_i) \leq I_2(B_i)$ for all $i \in \{1, \ldots, n\}$. Consequently, it is satisfied that $\hat{I}(\mathcal{B}) = \hat{I}(@[B_1, \ldots, B_n])$. Hence, by the unique homomorphic extension and the monotonicity of $@$, we have that

$$\begin{aligned}
\hat{I}_1(\mathcal{B}) &= \hat{I}_1(@[B_1, \ldots, B_n]) \\
&= \dot{@}\left[\hat{I}_1(B_1), \ldots, \hat{I}_1(B_n)\right] \\
&\leq \dot{@}\left[\hat{I}_2(B_1), \ldots, \hat{I}_2(B_n)\right] \\
&= \hat{I}_2(@[B_1, \ldots, B_n]) \\
&= \hat{I}_2(\mathcal{B})
\end{aligned}$$

□

From Propositions 3 and 4, and taking into account that $T_{\mathbb{P}}^{Q_\mu}(I)(A) = Q_\mu\left(C_I^A\right)$, we obtained the following result.

Corollary 1. *The immediate consequences operator $T_{\mathbb{P}}^{Q_\mu}$ is increasing in $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$.*

In this subsection, we have introduced a generalization of the immediate consequences operator through generalized quantifiers, which preserves the monotonicity property.

3.3. Towards the quantified fixed point semantics. The particular case of $T_{\mathbb{P}}^{Q_{\mu^1}}$

Once the new definition of immediate consequences operator has been introduced, the next step would be to study the semantics associated with the new immediate consequences operator $T_{\mathbb{P}}^{Q_\mu}$. Before that, for a better understanding in this section, the particular operator associated with the μ^1 measure (defined in Example 6) will be analyzed. This simple measure is related to the generalized quantifier that removes the greatest value of the given set, which is one of the simplest approaches for removing a possible noise or wrong recorded data.

First of all, the main goal now is to characterize the least fixed point of the operator $T_{\mathbb{P}}^{Q_{\mu^1}}$ as the least model of the program. Clearly, the definition of model must be changed in order to capture the philosophy of eliminating possible noise in the data.

Definition 14. (μ^1*-weak model*). An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is said to be a μ^1*-weak model* of a multi-adjoint logic program $\mathbb{P}$ if, for each $A \in \Pi$, all the rules with head A are satisfied by I, except possibly one of them, i.e., there can exist a rule $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}$ such that $\vartheta \not\leq \hat{I}(A \leftarrow_i \mathcal{B})$.

The first property introduces the characterization of the μ^1-weak model of a program $\mathbb{P}$ as the post-fixed points of the generalized definition of immediate consequences operator through the fuzzy measure μ^1.

Theorem 5. *Let $\mathbb{P}$ be a multi-adjoint logic program and I an interpretation of $\mathbb{P}$. Then, I is a μ^1-weak model of $\mathbb{P}$ if and only if $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$.*

Proof. Let $\{u_1, u_2, \ldots, u_n\}$ be the set of rules of $\mathbb{P}$ with head the propositional symbol $A \in \Pi$. We can consider a permutation π in this set in such a way that it is verified the following chain of inequalities

$$C_I^A\left(u_{\pi(n)}\right) \leq C_I^A\left(u_{\pi(n-1)}\right) \leq \ldots \leq C_I^A\left(u_{\pi(2)}\right) \leq C_I^A\left(u_{\pi(1)}\right) \tag{2}$$

Let I be a μ^1-weak model of $\mathbb{P}$. We are going to prove that $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$. As the interpretation I is a μ^1-weak model, then all the rules of the program, except possibly one of them, are satisfied by I. Suppose that I does not satisfy the rule $u_{\pi(i)}$, for some $i \in \{2, \ldots, n\}$. It is verified that $\vartheta \not\leq \hat{I}\left(A \leftarrow_i \mathcal{B}_{\pi(i)}\right)$ or equivalently using the adjoint property and the definition of $\hat{I}$, $\vartheta \dot{\&}_i \hat{I}\left(\mathcal{B}_{\pi(i)}\right) \not\leq I(A)$ and it is equivalent to $C_I^A\left(u_{\pi(i)}\right) \not\leq I(A)$.

On the other hand, the rule $u_{\pi(i-1)}$ must be satisfied by I. Therefore, it must be verified that $\vartheta \leq \hat{I}\left(A \leftarrow_i \mathcal{B}_{\pi(i-1)}\right)$, or equivalently, $\vartheta \dot{\&}_i \hat{I}\left(\mathcal{B}_{\pi(i-1)}\right) \leq I(A)$, which is equivalent to $C_I^A\left(u_{\pi(i-1)}\right) \leq I(A)$.

Consequently, by the order given in Expression (2) we have obtained that $C_I^A\left(u_{\pi(i)}\right) \leq C_I^A\left(u_{\pi(i-1)}\right) \leq I(A)$ and we have reached a contradiction with $C_I^A\left(u_{\pi(i)}\right) \not\leq I(A)$.

We can conclude that the only rule that might not be satisfied by the interpretation I is $u_{\pi(1)}$. In particular, $u_{\pi(2)}$ is verified by the interpretation I, i.e., we reach the following chain of equivalent inequalities

$$\begin{aligned} \vartheta &\leq \hat{I}\left(A \leftarrow_i \mathcal{B}_{\pi(2)}\right) \\ \vartheta \dot{\&}_i \hat{I}\left(\mathcal{B}_{\pi(2)}\right) &\leq I(A) \\ C_I^A\left(u_{\pi(2)}\right) &\leq I(A) \end{aligned}$$

Taking into account the order aforementioned and the choice of the fuzzy measure μ^1, we can obtain that

$$\begin{aligned} T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) &= Q_{\mu^1}\left(C_I^A\right) \\ &= \bigvee_{i=1}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1, \ldots, u_i\}\right) \\ &= \bigvee_{i=2}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1, \ldots, u_i\}\right) \\ &= C_I^A\left(u_{\pi(2)}\right) \leq I(A) \end{aligned}$$

Therefore, we have proved that $T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) \leq I(A)$, for all $A \in \Pi$, i.e., $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$.

Now, we prove the opposite implication. For this, consider an interpretation I satisfying $T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I$. First of all, $T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) = C_I^A\left(u_{\pi(2)}\right) \leq I(A)$. By the order given by π, it is verified for all $i \in \{2, \ldots, n\}$, that $C_I^A\left(u_{\pi(i)}\right) \leq I(A)$ and this expression is equivalent to $\vartheta \leq \hat{I}\left(A \leftarrow_i \mathcal{B}_{\pi(i)}\right)$. Therefore, all the rules except possibly one of them, which is the associate with $\pi(1)$, are satisfied by I. Consequently, I is a μ^1-weak model of the program $\mathbb{P}$. □

The following result focuses the attention to a particular μ^1-weak model and fixed point.

Proposition 5. *The interpretation* $M_{\mathbb{P}} : \Pi \longrightarrow [0, 1]$ *given by*

$$M_{\mathbb{P}} = \inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } \mu^1\text{-weak model of } \mathbb{P}\right\}$$

is a μ^1*-weak model of the program* $\mathbb{P}$*, which is the least* μ^1*-weak model of* $\mathbb{P}$ *and is such that* $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{Q_{\mu^1}}\right)$.

Proof. Notice that the infimum of the set $\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } \mu^1\text{-weak model of } \mathbb{P}\}$ exists because $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$ is a complete lattice.

Taking into account that $T_{\mathbb{P}}^{Q_{\mu^1}}$ is a monotonic mapping, similar to Proposition 1, it has a least fixed point that verifies the following expression [27]

$$\operatorname{lfp}\left(T_{\mathbb{P}}^{Q_{\mu^1}}\right) = \inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I\right\}$$

Moreover, by Theorem 5 we obtain that

$$\inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}^{Q_{\mu^1}}(I) \sqsubseteq I\right\} = \inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } \mu^1\text{-weak model of } \mathbb{P}\right\} = M_{\mathbb{P}}$$

In addition, by the definition of fixed point, we have that $T_{\mathbb{P}}^{Q_{\mu^1}}\left(\text{lfp}\left(T_{\mathbb{P}}^{Q_{\mu^1}}\right)\right) = \text{lfp}\left(T_{\mathbb{P}}^{Q_{\mu^1}}\right)$. Therefore, $\text{lfp}\left(T_{\mathbb{P}}^{Q_{\mu^1}}\right)$ is a μ^1-weak model of $\mathbb{P}$ from Theorem 5. Thus, $M_{\mathbb{P}}$ is a μ^1-weak model of $\mathbb{P}$, which is the least μ^1-weak model of $\mathbb{P}$. □

Finally, we reach the following generalization of Theorem 3.

Theorem 6. *Let $\mathbb{P}$ be a non-empty multi-adjoint logic program. Then, for some ordinal α, it is verified that $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{Q_{\mu^1}}\right) = T_{\mathbb{P}}^{\alpha}(\triangle)$.*

Proof. The proof of this result follows directly from Theorem 5 and Corollary 1. □

We conclude that this new definition of the immediate consequences operator, through generalized quantifiers, helps to discard possible noise in the data (as we show in Example 5) and consider other values in the computation of $T_{\mathbb{P}}$, besides the supremum, which makes it a beneficial advantage in different noisy data sets. However, it also presents a disadvantage related to those propositional symbols that are the head of a single rule. In this case, if we want to determine the value of $T_{\mathbb{P}}^{Q_{\mu^1}}$, we will reach the null value because of the non-existence of a second greatest value. With the aim of solving this problem, diverse possible solutions exist. One of them consists in trying to determine more rules with head the propositional symbol in question. Another possible solution is based on the consideration in Example 6 of a different fuzzy measure, such as the measure μ^2, defined as

$$\mu^2(D) = \begin{cases} 0 & \text{if } D = \varnothing \\ 0.8 & \text{if } D = \{u_i\} \text{ for all } i \in \{1, \ldots, n\} \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

The aim of considering this new fuzzy measure is to slightly impair the rule with the greatest value $C_I^A(u)$.

Finally, another possibility is to modify the given definition of $T_{\mathbb{P}}^{Q_{\mu}}$ (Definition 13). Next, a new definition of the immediate consequences operator through generalized quantifiers will be introduced with the goal of solving the problem generated by those propositional symbols which are head of a single rule. This new definition will offer more flexibility in the selection of the fuzzy measure. We will denote $\mathbb{P}_A$ the subset of rules of a program $\mathbb{P}$ with head a propositional symbol A.

Definition 15. (*M-quantified immediate consequences operator*). Let $\mathbb{P}$ be a multi-adjoint logic program, $M = \{\mu_A : \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi, \mathbb{P}_A \neq \varnothing\}$ a family of fuzzy measures invariant with respect to the cardinality. We define the *M-quantified immediate consequences operator obtained from the family M* as the mapping $T_{\mathbb{P}}^M : \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$ given by

$$T_{\mathbb{P}}^M(I)(A) = \begin{cases} Q^{\mu_A}(C_I^A) & \text{if} \quad \mathbb{P}_A \neq \varnothing \\ 0 & \text{if} \quad \mathbb{P}_A = \varnothing \end{cases} \tag{4}$$

for all $A \in \Pi$ and $I \in \mathcal{I}_{\mathfrak{L}}$.

Notice that the definition of $T_{\mathbb{P}}^M$ has been given by cases because generalized quantifiers are defined on non-empty finite universes and in this case the universe associated with the generalized quantifiers Q^{μ_A} is $\mathbb{P}_A$. Moreover, with this definition if $|\mathbb{P}_A| = 1$, then we can take $\mu = \mu^{\exists}$ and we obtain that $T_{\mathbb{P}}^M(I)(A) = Q^{\mu_A}(C_I^A) = Q^{\exists}(C_I^A)$, which avoids the problem aforementioned.

Example 7. Let us consider a new program $\mathbb{P}_2$ from Example 2 with the extra rule

$$\langle \texttt{flu} \leftarrow_{\text{G}} \texttt{diarrhoea} \wedge_{\text{Ł}} \texttt{moderate fever}, 0.6 \rangle$$

Clearly, if the main goal is to remove the possible noise presented in the data, we cannot consider the same fuzzy measure for rules whose head is `covid` and for those of `flu`. For example, if we consider the fuzzy measure μ^1 for both propositional symbols, all information associated with `flu` is removed and we obtain that $T_{\mathbb{P}_2}^{Q_{\mu_1}}(I)(\texttt{flu}) = 0$. Hence, Definition 15 should be considered. In this example, we consider the interpretation I defined in Example 5 and the family of fuzzy measures $M = \{\mu_{\texttt{covid}}, \mu_{\texttt{flu}}\}$ given by $\mu_{\texttt{covid}} = \mu^1$ and $\mu_{\texttt{flu}} = \mu^{\exists}$ is taken into account. Therefore,

$$T_{\mathbb{P}_2}^{M}(I)(\texttt{covid}) = T_{\mathbb{P}_2}^{Q_{\mu_{covid}}}(I)(\texttt{covid}) = 0.5$$
$$T_{\mathbb{P}_2}^{M}(I)(\texttt{flu}) = T_{\mathbb{P}_2}^{Q_{\mu_{flu}}}(I)(\texttt{flu}) = 0.6$$ □

The next result is really important because it shows that in some sense this new operator generalizes the one given in Definition 13.

Theorem 7. *Let $\mathbb{P}$ be a multi-adjoint logic program and μ a fuzzy measure invariant with respect to the cardinality. Then there exists a family $M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi\}$ of fuzzy measures invariant with respect to the cardinality, such that $T_{\mathbb{P}}^{\mu} = T_{\mathbb{P}}^{M}$.*

Proof. Consider the family of fuzzy measures invariant with respect to the cardinality $M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi, \mathbb{P}_A \neq \varnothing\}$, where $\mu_A = \mu_{|\mathcal{P}(\mathbb{P}_A)}$, for all $A \in \Pi$ and $\mathbb{P}_A \neq \varnothing$. Then, denoting $|\mathbb{P}_A| = n_A$ and $|\mathbb{P}| = n$, and π_A and π the corresponding permutations, the following equalities hold for those propositional symbols whose $\mathbb{P}_A$ is not empty and for all interpretation $I \in \mathcal{I}_{\mathfrak{L}}$.

$$\begin{aligned}
T_{\mathbb{P}}^{M}(I)(A) &= Q^{\mu_A}\left(C_I^A\right) \\
&= \bigvee_{i=1}^{n_A} C_I^A\left(u_{\pi_A(i)}\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \\
&= \bigvee_{i=1}^{n_A} C_I^A\left(u_{\pi_A(i)}\right) \otimes \mu_{|\mathcal{P}(\mathbb{P}_A)}\left(\{u_1, \ldots, u_i\}\right) \\
&\overset{(1)}{=} \bigvee_{i=1}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu\left(\{u_1, \ldots, u_i\}\right) \\
&= T_{\mathbb{P}}^{Q_\mu}(I)(A)
\end{aligned}$$

where (1) holds because $C_I^A\left(u_{\pi(i)}\right) = 0$ for all rule $u_{\pi(i)}$ non-headed by A. If $\mathbb{P}_A = \varnothing$, then $T_{\mathbb{P}}^{M}(I)(A) = \bigvee_{D \in \mathcal{P}(\mathbb{P})} \left(\bigwedge_{u \in D} C_I^A(u) \right) \otimes \mu(D) = \bigvee_{D \in \mathcal{P}(\mathbb{P})} \left(\bigwedge_{u \in D} 0 \right) \otimes \mu(D) = \bigvee_{D \in \mathcal{P}(\mathbb{P})} 0 \otimes \mu(D) = 0 = T_{\mathbb{P}}^{Q_\mu}(I)(A)$, for all interpretation $I \in \mathcal{I}_{\mathfrak{L}}$. □

The following example shows how the family is obtained from a given measure.

Example 8. Returning to Example 7, we can consider the fuzzy measure $\tau \colon \mathcal{P}(\mathbb{P}) \to [0,1]$ defined for all $D \in \mathcal{P}(\mathbb{P})$ as

$$\tau(D) = \begin{cases} 0 & \text{if } D = \varnothing \\ \dfrac{1}{2} & \text{if } D \neq \varnothing,\ D \neq \mathcal{P}(\mathbb{P}) \\ 1 & \text{if } D = \mathcal{P}(\mathbb{P}) \end{cases} \tag{5}$$

In this case, we obtain that $T_{\mathbb{P}}^{Q_\tau}(I)(\texttt{covid}) = 0.405$ and $T_{\mathbb{P}}^{Q_\tau}(I)(\texttt{flu}) = 0.3$.

From this measure, taking into account Proposition 7, we consider the restriction of τ to $\mathcal{P}(\mathbb{P}_{\texttt{covid}})$ and $\mathcal{P}(\mathbb{P}_{\texttt{flu}})$, that is $\tau_{\texttt{covid}} \colon \mathcal{P}(\mathbb{P}_{\texttt{covid}}) \to [0,1]$, $\tau_{\texttt{flu}} \colon \mathcal{P}(\mathbb{P}_{\texttt{flu}}) \to [0,1]$, defined as

$$\tau_{\texttt{covid}}(D_c) = \begin{cases} 0 & \text{if} \quad D_c = \varnothing \\ \dfrac{1}{2} & \text{if} \quad |D_c| \in \{1,2,3\} \end{cases} \qquad \tau_{\texttt{flu}}(D_f) = \begin{cases} 0 & \text{if} \quad D_f = \varnothing \\ \dfrac{1}{2} & \text{if} \quad |D_f| = 1 \end{cases}$$

for all $D_c \in \mathcal{P}(\mathbb{P}_{\texttt{covid}})$, $D_f \in \mathcal{P}(\mathbb{P}_{\texttt{flu}})$. Clearly, $M = \{\tau_{\texttt{covid}}, \tau_{\texttt{flu}}\}$ is a family of fuzzy measures which are not normalized, satisfying that

$$T_{\mathbb{P}}^{M}(I)(\texttt{covid}) = T_{\mathbb{P}}^{Q_{\tau_{covid}}}(I)(\texttt{covid}) = 0.405 = T_{\mathbb{P}}^{Q_\tau}(I)(\texttt{covid})$$
$$T_{\mathbb{P}}^{M}(I)(\texttt{flu}) = T_{\mathbb{P}}^{Q_{\tau_{flu}}}(I)(\texttt{flu}) = 0.3 = T_{\mathbb{P}}^{Q_\tau}(I)(\texttt{flu})$$

Due to restrictions of the original mapping that are considered, the main properties, such as the monotonicity, are preserved. □

The following result shows that this new operator is also increasing.

Proposition 6. *Let $M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi,\ \mathbb{P}_A \neq \varnothing\}$ be a family of fuzzy measures invariant with respect to the cardinality. Then, the immediate consequences operator $T_{\mathbb{P}}^{M}$ introduced in Definition 4 is increasing in $\mathcal{I}_{\mathfrak{L}}$.*

Proof. Let $I_1, I_2 \in \mathcal{I}_{\mathfrak{L}}$ be two interpretations such that $I_1 \sqsubseteq I_2$, then for the monotonicity of the mapping C_I^A given by Proposition 4 it is verified that $C_{I_1}^A \leq C_{I_2}^A$. Moreover, due to the fact that μ_A is a fuzzy measure invariant with respect to the cardinality for all $A \in \Pi$ with $\mathbb{P}_A \neq \varnothing$, by Proposition 3, we have that Q^{μ_A} is increasing and so, $T_{\mathbb{P}}^{M}(I_1)(A) = Q^{\mu_A}(C_{I_1}^A) \leq Q^{\mu_A}(C_{I_2}^A) = T_{\mathbb{P}}^{M}(I_2)(A)$, for all $A \in \Pi$, with $\mathbb{P}_A \neq \varnothing$. Since $T_{\mathbb{P}}^{M}(I_1)(A) = T_{\mathbb{P}}^{M}(I_2)(A)$ trivially holds for all $A \in \Pi$, with $\mathbb{P}_A = \varnothing$, we obtain that $T_{\mathbb{P}}^{M}$ is increasing. □

Therefore, we have introduced a new definition of quantified immediate consequences operator, which generalizes the $T_{\mathbb{P}}^{Q_\mu}$ operator and solves the problem related to those propositional symbols head of a single rule. Furthermore, it provides an extra level of flexibility for considering one fuzzy measure for each propositional symbol. Clearly, by Proposition 6, all the results proved in the rest of the paper for $T_{\mathbb{P}}^{M}$ will be true for $T_{\mathbb{P}}^{Q_\mu}$, since the normalization property is not fundamental for proving them. In the following subsection, we will study the semantics of this operator $T_{\mathbb{P}}^{M}$.

3.4. Semantics of the $T_{\mathbb{P}}^{M}$ with a general family of fuzzy measures M

In this subsection, we are going to define the semantics of a multi-adjoint logic program from the immediate consequences operator obtained through generalized quantifiers with a family of fuzzy measures M (Definition 15).

Due to the complexity of providing a semantics description of an arbitrary generalized quantifier, it is difficult to give a natural definition of model. Consequently and taking into account the results presented previously (the characterization of μ^1-weak models as post-fixed points of the $T_{\mathbb{P}}^{Q_{\mu^1}}$ and the determination of the least μ^1-weak model of the program as the least fixed point of the $T_{\mathbb{P}}^{Q_{\mu^1}}$), we can introduce the following definition of model associated with a family of fuzzy measures M.

Definition 16. (*M-model*). Let $M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi,\ \mathbb{P}_A \neq \varnothing\}$ be a family of fuzzy measures invariant with respect to the cardinality. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a *M-model* of a multi-adjoint logic program $\mathbb{P}$ if it is a post-fixed point of $T_{\mathbb{P}}^{M}$.

Taking into account Remark 3, we cannot define M-models as the interpretations greater or equal than the least fixed point of $T_{\mathbb{P}}^{M}$, but as the post-fixed points of $T_{\mathbb{P}}^{M}$, considering Theorem 1.

It is straightforwardly satisfied that the set of M-models with the order previously defined, is a complete lattice. By the monotonicity of $T_{\mathbb{P}}^{M}$ given by Proposition 6, it has a least fixed point $\mathrm{lfp}\big(T_{\mathbb{P}}^{M}\big)$ by the general result of Proposition 1 given in [27]. Furthermore, it verifies that

$$\mathrm{lfp}\left(T_{\mathbb{P}}^{M}\right) = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}^{M}(I) \sqsubseteq I\}$$

We can characterize the least M-model as the least fixed point of the operator $T_{\mathbb{P}}^{M}$ using Definition 16. With this goal, we present the following results.

Proposition 7. *The interpretation* $M_{\mathbb{P}} : \Pi \longrightarrow [0,1]$ *given by*

$$M_{\mathbb{P}} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } M\text{-model of } \mathbb{P}\}$$

is a M*-model of* $\mathbb{P}$*, which is the least* M*-model of* $\mathbb{P}$ *and* $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{M}\right)$.

Proof. The proof is analogous to the one given in Proposition 5. □

Finally, we reach the generalization of Theorem 3 to this new framework of the immediate consequences operator.

Theorem 8. *Let* $\mathbb{P}$ *be a multi-adjoint logic program. Then, for some ordinal* α*, it is verified that* $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{M}\right) = \left(T_{\mathbb{P}}^{M}\right)^{\alpha}(\triangle)$.

Proof. The proof of this result follows directly from Definition 16 and Corollary 1. □

In the following example, we can see how the least M-model can be obtained in this particular case of $T_{\mathbb{P}}^{M^1}$, where

$$M^1 = \left\{\mu_{\texttt{covid}} = \mu^1,\ \mu_A = \mu^{\exists} \text{ for all } A \in \Pi,\ A \neq \texttt{covid}\right\}$$

Example 9. Taking into account Example 2, we are going to show how to calculate the value of the immediate consequences operator and the least model of the program $\mathbb{P}_1$. From Theorem 8, the least model of $\mathbb{P}_1$ is obtained through iterating the $T_{\mathbb{P}_1}^{M^1}$ starting from the least interpretation $\triangle$. From the unique homomorphic extension, we can obtain

$$\begin{aligned} T_{\mathbb{P}_1}^{M^1}(\triangle)(\texttt{covid}) &= \bigvee_{i=1}^{3} C_{\triangle}^{\texttt{covid}}\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1,\ldots,u_i\}\right) \\ &= \sup\{0,0,0\} \\ &= 0 \end{aligned}$$

$$\begin{aligned} \left(T_{\mathbb{P}_1}^{M^1}\right)^2(\triangle)(\texttt{covid}) &= \bigvee_{i=1}^{3} C_{T_{\mathbb{P}_1}^{M^1}(\triangle)}^{\texttt{covid}}\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1,\ldots,u_i\}\right) \\ &= \bigvee\{0.56 \otimes 0, 0.54 \otimes 1, 0.3 \otimes 1\} \\ &= \bigvee\{0, 0.54, 0.3\} \\ &= 0.54 \end{aligned}$$

In the following table we show the iteration of $T_{\mathbb{P}_1}^{M^1}$ from the least interpretation $\triangle$. Note that, in this simple example without loops (which is possible in general), the least fixed point is reached in the second iteration; hence Theorem 3 entails that the least model of $\mathbb{P}_1$ corresponds to $\left(T_{\mathbb{P}_1}^{M^1}\right)^2(\triangle)$.

	$\triangle$	$T_{\mathbb{P}_1}^{M^1}(\triangle)$	$\left(T_{\mathbb{P}_1}^{M^1}\right)^2(\triangle)$	$\left(T_{\mathbb{P}_1}^{M^1}\right)^3(\triangle)$
`moderate fever`	0	0.6	0.6	0.6
`cough`	0	0.8	0.8	0.8
`diarrhoea`	0	0.6	0.6	0.6
`headache`	0	0.7	0.7	0.7
`waist pain`	0	0.3	0.3	0.3
`covid`	0	0	0.54	0.54

□

In conclusion, the immediate consequences operator has been conveniently generalized through generalized quantifiers, which will provide greater flexibility in mathematical modeling and resolution of different problems for obtaining information from databases and for data processing. It is important to bear in mind that the measure is fundamental to computing the results and that not every measure μ is good to generalize the $T_{\mathbb{P}}$. If we consider the minimum measure given at Example 4, we reach the dual definition of the immediate consequences operator. Regarding the selection of the fuzzy measure and, therefore, the quantifier, the best is to consider those that can ensure us values close to the supremum, so as not to significantly distance ourselves from the original definition of the $T_{\mathbb{P}}$. This goal is studied in the following section.

4. Generalized orness measure

In the previous section we have modified the definition of the immediate consequences operator to solve the possible presence of noise in the data. However, we cannot lose sight of the definition of the immediate consequences operator, i.e., it is advisable to stay with quantifiers close to the existential one (modeled by the supremum), which is the one used by the immediate consequences operator.

With the aim of classifying different quantified immediate consequences operators, depending on the considered measure, we introduce an adaptation of the measure known as *orness* in the framework of ordered weighted averaging (OWA) operators [29,17].

Definition 17. (μ-*orness*). Given a fuzzy measure μ invariant with respect to the cardinality, we define the μ-*orness measure* associated with μ as the mapping orness: $\Sigma(U) \longrightarrow [0,1]$ defined as

$$\text{orness}(\mu) = \frac{2}{n(n-1)} \sum_{i=1}^{n} (n-i)\mu(\{u_1, \ldots, u_i\})$$

where $U = \{u_1, \ldots, u_n\}$ is an ordered set verifying $u_n \leq u_{n-1} \leq \ldots \leq u_2 \leq u_1$ and $\Sigma(U) = \{\mu : U \longrightarrow [0,1] \mid \mu$ is a fuzzy measure invariant with respect to cardinality$\}$.

The correspondent addition of the values $(n-i)\mu^{\exists}(\{u_1, \ldots, u_i\})$, with $i \in \{1, \ldots, n\}$, is $\frac{n(n-1)}{2}$. Consequently, in order to obtain $\text{orness}(\mu^{\exists}) = 1$, we must multiply it by $\frac{2}{n(n-1)}$. In addition, for other fuzzy measures μ, the sum $(n-i)\mu(\{u_1, \ldots, u_i\})$ will be always less than that obtained for the existential measure, which ensures that the orness value is less than 1. Next, we introduce an example in which we calculate the value of the orness associated with some important examples of fuzzy measures.

Example 10. The following conditions hold.

a) For $\mu = \mu^{\forall}$, given in Example 4, it is verified that $\text{orness}\left(\mu^{\forall}\right) = 0$.
b) For $\mu = \mu^{\exists}$, given in Example 4, it is verified that $\text{orness}\left(\mu^{\exists}\right) = 1$.
c) The fuzzy measure μ^3 defined as

$$\mu^3(D) = \begin{cases} 1 & \text{if} \quad D = U \\ \frac{1}{2} & \text{if} \quad D \neq U \text{ and } D \neq \varnothing \\ 0 & \text{if} \quad D = \varnothing \end{cases}$$

verifies that $\text{orness}(\mu^3) = 1/2$.
d) Considering the fuzzy measure μ^1, defined by Expression (1), it holds that $\text{orness}(\mu^1) = \frac{n-2}{n}$. Clearly, $\frac{1}{2} \leq \text{orness}(\mu^1)$, if $4 \leq n$ and the orness of μ^1 approaches 1, when the number of rules increase. Notice that in this case we are considering the universe $U = \mathbb{P}$, due to the fact that the fuzzy measure μ^1 is defined on a multi-adjoint logic program.
e) The fuzzy measure μ^2, defined by Expression (3), verifies $\text{orness}(\mu^2) = \frac{n-0.4}{n}$. Therefore, $\text{orness}(\mu^2) \geq \frac{1}{2}$ if and only if $n \geq 1$. Consequently, the use of the fuzzy measure μ^2 is a good generalization of the immediate consequences operator.

As we have already commented, when we are generalizing the immediate consequences operator, we are interested in considering a great orness value. For this, we present some results that consider fuzzy measures with a great orness.

Theorem 9. *Let* $\mu\colon \mathcal{P}(U) \to [0,1]$ *and* $\mu'\colon \mathcal{P}(U) \to [0,1]$ *two fuzzy measures, where* $U = \{u_1, \ldots, u_n\}$ *such that*

- $\mu_i = \mu'_i$, *for* $i \neq j$ *and* $i \neq k$,
- $\mu_j = \mu'_j + \delta$,
- $\mu_k = \mu'_k - \delta$

where $\mu_i = \mu(\{u_1, \ldots, u_i\})$ *and* $\delta > 0$ *and* $j < k$. *Then, orness* $\left(\mu'\right) <$ *orness* (μ).

Proof. From Definition 17,

$$\begin{aligned}\text{orness}(\mu) &= \frac{2}{n(n-1)}\sum_{i=1}^{n}(n-i)\mu_i \\ &= \frac{2}{n(n-1)}\sum_{i=1}^{n}(n-i)\mu'_i + \frac{2}{n(n-1)}\left((n-j)\delta - (n-k)\delta\right)\end{aligned}$$

Consequently, $\text{orness}(\mu) = \text{orness}\left(\mu'\right) + \frac{2}{n(n-1)}\delta(k-j)$. As $k > j$, it holds that $\text{orness}\left(\mu'\right) < \text{orness}(\mu)$. □

Example 11. Consider the fuzzy measure μ^2, defined by Expression (3), and the measure defined as

$$\mu^4(D) = \begin{cases} 0 & \text{if } D = \varnothing \\ 0.9 & \text{if } D = \{u_i\} \text{ for all } i \in \{1, \ldots, n\} \\ 0.9 & \text{if } D = \left\{u_i, u_j\right\} \text{ for all } i, j \in \{1, \ldots, n\}, i \neq j \\ 1 & \text{otherwise} \end{cases}$$

From Theorem 9, $\text{orness}(\mu^2) \leq \text{orness}(\mu^4)$. It can be proved that the orness measure for this case is $\text{orness}(\mu^4) = \frac{1.8}{n} + \frac{(n-2)(n-1.2)}{n(n-1)}$. □

In the following result, we present a sufficient condition to ensure an orness value higher than $\frac{1}{2}$.

Proposition 8. *Let* $\mu\colon U \to [0,1]$ *be a fuzzy measure defined as*

$$\mu(D) = \begin{cases} 1 & \text{if } D = U \\ \alpha & \text{if } D \neq U \text{ and } D \neq \varnothing \\ 0 & \text{if } D = \varnothing \end{cases}$$

where $\alpha \in [0,1]$. *If* $\alpha > \frac{1}{2}$, *then* $orness(\mu) > \frac{1}{2}$.

Proof. From Definition 17,

$$\begin{aligned}\text{orness}(\mu) &= \frac{2}{n(n-1)}\sum_{i=1}^{n}(n-i)\mu_i \\ &= \frac{2}{n(n-1)}\sum_{i=1}^{n-1}\alpha(n-i) \\ &= \alpha\end{aligned}$$

□

The following example will make a comparison among different measures used in the quantified immediate consequences operator.

Example 12. Given the multi-adjoint program given in Example 3 and the interpretation defined in Example 5, we obtain the following values for the quantified immediate consequences operator $T_{\mathbb{P}_1}^{Q_\mu}$ and the orness value, with respect to different measures.

	covid	$\mu^\exists$	μ^1	μ^2	μ^3	μ^4	$\mu^\forall$
$n=3$	$T_{\mathbb{P}_1}^{Q_\mu}(I)$	0.81	0.5	0.648	0.405	0.729	0.49
	orness	1	0.33	0.87	0.5	0.9	0

As we argued previously, the orness of the fuzzy measures depends on the number of rules for every propositional symbol. Consider now the multi-adjoint program given in Example 3 together with the following rules

$$\texttt{covid} \xleftarrow{0.8}_{\mathrm{P}} \texttt{loss of smell and taste}$$
$$\texttt{covid} \xleftarrow{0.4}_{\mathrm{G}} \texttt{sore throat} \wedge_{\mathrm{Ł}} \texttt{muscle pain}$$
$$\texttt{covid} \xleftarrow{0.6}_{\mathrm{G}} \texttt{stuffy and runny nose} \wedge_{\mathrm{Ł}} \texttt{shortness of breath}$$
$$\texttt{covid} \xleftarrow{0.6}_{\mathrm{P}} \texttt{fatigue}$$

and the interpretation defined in Example 5 extended on the new propositional symbols as follows: $I(\texttt{sore throat}) = 0.65$, $I(\texttt{stuffy and runny nose}) = 0.8$, $I(\texttt{muscle pain}) = 0.55$, $I(\texttt{fatigue}) = 0.55$, $I(\texttt{loss of smell and taste}) = 0.75$, $I(\texttt{shortness of breath}) = 0.6$. Then, we obtain

	covid	$\mu^\exists$	μ^1	μ^2	μ^3	μ^4	$\mu^\forall$
$n=7$	$T_{\mathbb{P}_3}^{Q_\mu}(I)$	0.81	0.6	0.648	0.405	0.729	0.2
	orness	1	0.71	0.94	0.5	0.95	0

As a consequence, the orness of the fuzzy measures increases and more proper measures arise to be considered instead of the existential one. Thus, it is important to have the possibility of considering a different measure for each propositional symbol, as Definition 15 allows. □

Hence, we are going now to adapt the new definition of orness measure to the case of $T_{\mathbb{P}}^M$ (Definition 15). We are interested in choosing a family of fuzzy measures M with values close to the greatest one. For that, we present a new definition of orness measure adapted to this case.

Definition 18. (*M-orness*). Given a family of fuzzy measures invariant with respect to the cardinality $M = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi,\ \mathbb{P}_A \neq \varnothing\}$, we define the *M-orness measure* associated with the family M as the mapping

$$\text{orness} \colon \Sigma(\mathbb{P}_{A_1}) \times \ldots \times \Sigma(\mathbb{P}_{A_n}) \longrightarrow [0,1]$$

defined as follows

$$\text{orness}(M) = \inf\{\text{orness}(\mu_A) \mid \mu_A \in M\}$$

where $\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1]$, $\text{orness}(\mu_A)$ is given by Definition 17 and $\Sigma(U) = \{\mu \colon U \longrightarrow [0,1] \mid \mu \text{ is a fuzzy measure invariant with respect to cardinality}\}$. Notice that if $|\mathbb{P}_A| < 2$, then the fuzzy measure is the existential with orness value of 1, which does not affect the calculation of the infimum.

Example 13. Given a program $\mathbb{P}$, if we consider the family of fuzzy measures $M_1 = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi,\ \mathbb{P}_A \neq \varnothing\}$, where μ_A are defined for each $A \in \mathbb{P}$ as

$$\mu_A = \begin{cases} \mu^\exists & \text{if} \quad |\mathbb{P}_A| \in \{1,2\} \\ \mu^1 & \text{if} \quad |\mathbb{P}_A| > 2 \end{cases} \tag{6}$$

and μ^1 is the fuzzy measure defined in Expression (1), we have in general that orness$(\mu_A) < 1/2$. As we commented in Example 10(d), orness$(\mu^1) = \frac{k-2}{k}$ and so, if a propositional symbol is the head of three rules, then orness$(\mu_A) = 1/3$. Hence, if there is any propositional symbol that has less than 5 rules associated with it, then the orness value will be less than $\frac{1}{2}$ and so, the orness of the measure family M will be very different from the orness of the original $T_{\mathbb{P}}$, which is 1. Thus, this family is not appropriate for the generalization of the immediate consequences operator. In order to solve this drawback, we can adapt for instance the definition of μ_A given by Expression (6) to the following

$$\mu_A = \begin{cases} \mu^{\exists} & \text{if} \quad |\mathbb{P}_A| \in \{1,2,3,4\} \\ \mu^1 & \text{if} \quad |\mathbb{P}_A| > 4 \end{cases}$$

This new definition ensures that the orness value will be at least of 0.6, being closer to 1. In the future, a proper threshold of orness in a measure family will be studied. □

5. Conclusions and future work

This paper has been focused on improving the behaviour of the immediate consequences operator on datasets with imprecise data. We have presented the problematic generated by the definition of the $T_{\mathbb{P}}$ through the supremum of a set of values and, consequently, we have given a generalization of this operator based on generalized quantifiers. Furthermore, interesting properties have been proved such as the monotonicity. The semantics has been analyzed in a particular case and it has been translated into the general case providing a new definition of model, called M-model. Moreover, we have shown that, as usual, the least M-model can be computed as the least fixed point of $T_{\mathbb{P}}^{Q_M}$.

We have also exposed that not every fuzzy measure is convenient in order to solve the problem of the possible presence of noise in the data nor to generalize the $T_{\mathbb{P}}$. Hence, the definition of orness measure given in the framework of OWA operators has been adapted to be applied to fuzzy measures. This new definition allows us to be sure that the considered fuzzy measures are close to the supremum operator and so, the new definition of immediate consequences operator is also close to the original one.

In the future, more properties of the new quantified immediate consequences operator will be studied. Among them, we will pay special attention to the continuity, since this property is fundamental for ensuring the reaching of the least fixpoint in countable iterations [7]. Moreover, the new definitions and results will be extended to complete lattices and the comparison with the use of alternative definitions, such as the one given in [23], will be studied.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] P. Baldi, C. Fermüller, M. Hofer, On fuzzification mechanisms for unary quantification, Fuzzy Sets Syst. 388 (2020) 06.
[2] M. Calvo-Flores, M. Ruiz, D. Sánchez, M. Vila, Fuzzy quantification: a state of the art, Fuzzy Sets Syst. 242 (2014) 05.
[3] N. Cao, M. Holčapek, M. Štěpnička, Extensions of fuzzy relational compositions based on generalized quantifiers, Fuzzy Sets Syst. 339 (2018) 73–98, Theme: Algebra and Set Theory.
[4] N. Cao, M. Stepnicka, M. Burda, Fuzzy quantifiers and compositions of partial fuzzy relations employing dragonfly algebras, in: 2019 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2019, New Orleans, LA, USA, June 2019, pp. 23–26, IEEE, 2019, pp. 1–6.
[5] M. Cornejo, J. Díaz-Moreno, J. López-Rodríguez, J. Medina, Quantified concept-forming operators 2668 (2020) 273–280.
[6] M.E. Cornejo, J.C. Díaz-Moreno, J. Medina, Generalized quantifiers in formal concept analysis, J. Comput. Appl. Math. 404 (2022) 113772.
[7] C.V. Damásio, L.M. Pereira, Monotonic and residuated logic programs, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01, in: Lecture Notes in Artificial Intelligence, vol. 2143, 2001, pp. 748–759.
[8] M. Delgado, D. Sánchez, M.A. Vila, Fuzzy cardinality based evaluation of quantified sentences, Int. J. Approx. Reason. 23 (1) (2000) 23–66.
[9] D. Dubois, H. Prade, A. Rico, Residuated variants of Sugeno integrals: towards new weighting schemes for qualitative aggregation methods, Inf. Sci. 329 (2016) 765–781, Special issue on Discovery Science.

[10] A. Dvořák, M. Holčapek, L-fuzzy quantifiers of type ⟨1⟩ determined by fuzzy measures, Fuzzy Sets Syst. 160 (23) (2009) 3425–3452, Theme: Computer Science.
[11] A. Dvořák, M. Holčapek, Fuzzy measures and integrals defined on algebras of fuzzy subsets over complete residuated lattices, Inf. Sci. 185 (1) (2012) 205–229.
[12] A. Dvořák, M. Holčapek, On semantic properties of fuzzy quantifiers over fuzzy universes: restriction and living on, in: M.-J. Lesot, S. Vieira, M.Z. Reformat, J.P. Carvalho, A. Wilbik, B. Bouchon-Meunier, R.R. Yager (Eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer International Publishing, Cham, 2020, pp. 173–186.
[13] M.v. Emden, R. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (4) (1976) 733–742.
[14] C.G. Fermüller, C. Roschger, Randomized game semantics for semi-fuzzy quantifiers, Log. J. IGPL 22 (3) (2014) 413–439.
[15] I. Glöckner, Fuzzy Quantifiers: A Computational Theory, Studies in Fuzziness and Soft Computing, Springer, 2006.
[16] P. Julián-Iranzo, C. Rubio-Manzano, A sound and complete semantics for a similarity-based logic programming language, Fuzzy Sets Syst. 317 (2017) 1–26, Theme: Logic and Computer Science.
[17] A. Kishor, A.K. Singh, N.R. Pal, Orness measure of Owa operators: a new approach, IEEE Trans. Fuzzy Syst. 22 (4) (2014) 1039–1045.
[18] G. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic, Vol. 4, Prentice Hall, New Jersey, 1995.
[19] R. Kowalski, Predicate logic as programming language, Inf. Process. 74 (1974) 569–574.
[20] J. Lloyd, Foundations of Logic Programming, Springer Verlag, 1987.
[21] J. Medina, M. Ojeda-Aciego, P. Vojtáš, Multi-adjoint logic programming with continuous semantics, in: Logic Programming and Non-Monotonic Reasoning, LPNMR'01, in: Lecture Notes in Artificial Intelligence, vol. 2173, 2001, pp. 351–364.
[22] J. Medina, M. Ojeda-Aciego, P. Vojtáš, Similarity-based unification: a multi-adjoint approach, Fuzzy Sets Syst. 146 (2004) 43–62.
[23] V. Novák, A formal theory of intermediate quantifiers, Fuzzy Sets Syst. 159 (10) (2008) 1229–1246, Mathematical and Logical Foundations of Soft Computing.
[24] M. Štěpnička, N. Cao, M. Burda, A. Dolný, S. Ožana, The concept of unavoidable features in fuzzy relational compositions, Knowl.-Based Syst. 196 (2020) 105785.
[25] M. Štěpnička, M. Holčapek, Fuzzy relational compositions based on generalized quantifiers, in: A. Laurent, O. Strauss, B. Bouchon-Meunier, R.R. Yager (Eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer International Publishing, Cham, 2014, pp. 224–233.
[26] U. Straccia, M. Ojeda-Aciego, C.V. Damásio, On fixed-points of multivalued functions on complete lattices and their application to generalized logic programs, SIAM J. Comput. 38 (5) (2009) 1881–1911.
[27] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pac. J. Math. 5 (2) (1955) 285–309.
[28] M.H. van Emden, Quantitative deduction and its fixpoint theory, J. Log. Program. 1 (1986) 37–53.
[29] R.R. Yager, Families of OWA operators, Fuzzy Sets Syst. 59 (2) (1993) 125–148.

Capítulo 4

Alcance computacional de consecuencias cuantificadas a partir de datos iniciales imperfectos

Título:	Computational reaching of quantified consequences from imperfect initial data
Autores:	Jesús Medina, José Antonio Torné-Zambrano
Revista:	Mathematical Methods in the Applied Sciences
Factor de impacto JCR:	2,1. Posición 53 de 332 (Q1)
Año:	2024
Categoría:	Mathematics, Applied
Publicación:	27 de septiembre de 2024
DOI:	10.1002/mma.10542

Received: 19 April 2024 | Revised: 7 August 2024 | Accepted: 27 September 2024

DOI: 10.1002/mma.10542

RESEARCH ARTICLE

WILEY

Computational reaching of quantified consequences from imperfect initial data

Jesús Medina | **José Antonio Torné-Zambrano**

Department of Mathematics, University of Cádiz, Cádiz, Spain

Correspondence
Jesús Medina, Department of Mathematics, University of Cádiz, Cádiz, Spain.
Email: jesus.medina@uca.es

Communicated by: J. Vigo-Aguiar

Funding information
FEDER, UE; European Union NextGenerationEU/PRTR; MCIN/AEI/10.13039/501100011033, Grant/Award Number: TED2021-129748B-I00; MICIU/AEI/10.13039/501100011033, Grant/Award Number: PID2019-108991GB-I00 and PID2022-137620NB-I00

Usually, it is not direct to translate theoretical results to practice. One of the challenges is the real computation of solutions to proposed problems, when infinite numbers (such as real numbers or the unit interval) are considered. The same is true when consequences (information) from a data set modeled by logical rules need to be obtained. In this framework, this information is obtained from the immediate consequences operator and ensuring its continuity is fundamental for making sure the computation of this information in a countable number of iterations. Recently, a versatile operator using generalized quantifiers has been introduced. This note will advance in the knowledge introducing necessary and sufficient conditions to ensure the continuity of this new flexible operator.

KEYWORDS

fixed-point semantics, fuzzy sets, generalized quantifiers, immediate consequences operator

MSC CLASSIFICATION

03E72, 94D05, 03C80, 68N17

1 | INTRODUCTION

The resolution inference rule introduced by Robinson in the sixties [1] was fundamental to use logic as a programming language, giving rise to logic programming [2]. In this framework, a set of logic rules (program) is designed to represent knowledge about (a part of) the world, with the main goal of facilitating the reasoning on it to, for instance, decision-support systems. For this primary goal, its least model provides the most accurate information about the program (semantics) that allows it to be more precise to support the user in decision making. This theory has been enriched with the use of fuzzy sets, which have increased the flexibility of different approaches [3–9]. From a practical point of view, there has been also implementations of fuzzy logic programming through FASILL [10] or Bousi-Prolog [11, 12], which has been recently applied to different areas of knowledge such as cloud computing [13], social networks [14], SAT/SMT techniques [15], neural networks [16], explainable machine learning [17], genome-scale metabolic models [18], text classification [19], knowledge discovery [20], decision making [21], and semantic web [22].

In the framework of this paper, the information obtained from a logic program is defined through a fixed-point semantics, in which the least model of the program is the least fixed-point of the immediate consequences operator $T_{\mathbb{P}}$ and so it can be obtained iterating the $T_{\mathbb{P}}$ starting from the least interpretation $\vartriangle$. The immediate consequences operator is defined from the supremum of a set of values [23]. The strict nature of the supremum can cause that the presence of noise in the data alters the final result of the computation, as it can be seen in [24, Example 5], from an illustrative point of view. For this reason, in [24], the definition of the quantified immediate consequences operator $T^{\mathbb{M}}_{\mathbb{P}}$ was introduced, which

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.
© 2024 The Author(s). Mathematical Methods in the Applied Sciences published by John Wiley & Sons Ltd.

generalizes the definition of the $T_{\mathbb{P}}$ through a family of fuzzy measures M, based on generalized quantifiers [25, 26] and defined on the multi-adjoint setting [23]. The multi-adjoint approach presents several advantages that allow the representation of more complex systems [5, 27]. For example, the use of several adjoint pairs in the construction of logic programs is allowed, as well as less restrictive properties in the considered connectives [27].

Therefore, this generalization provides a more flexible and versatile operator that, although the use of quantifiers significantly changes the definition of the operator, it preserves interesting properties, such as the monotonicity, which ensures that if the confidence factor of the initial data is greater, then the value obtained for the conclusions is also greater [24]. However, its continuity has not been studied yet. This property ensures that small changes in the knowledge base lead to small variations in the obtained conclusions, providing stability and predictability to the reasoning process of the system.

Moreover, in [28], Damásio and Pereira state that if the operator $T_{\mathbb{P}}$ is not continuous, then more than a countable number of iterations may be needed to reach the least fixed point and, consequently, the least model of the program. Therefore, the continuity is one of the most desirable properties for the new definition of the immediate consequences operator $T^{\mathrm{M}}_{\mathbb{P}}$. Thus, this paper will establish necessary and sufficient conditions to ensure the continuity of $T^{\mathrm{M}}_{\mathbb{P}}$, extending the results given in [23].

The structure of the paper is as follows: In Section 2, some preliminaries results are recalled. In particular, the generalization of the immediate consequences operator based on generalized quantifiers is recalled. Section 3 presents the study of the continuity of the $T^{\mathrm{M}}_{\mathbb{P}}$ with the main results of the paper. Finally, some conclusions and future work are introduced in Section 4.

2 | PRELIMINARIES

This section will recall essential definitions considered in the paper. The definitions and results of multi-adjoint logic programming and generalized quantifiers used in this paper can be seen in [24].

The following result ensures that a monotonic mapping on a complete lattice has a least fixed point and a greatest one. This proposition is a weak result of a theorem given by Tarski in 1955 [29]. More details on fixed point theory can be seen in [2].

Proposition 1. ([29]) *Let $\langle L, \preccurlyeq\rangle$ be a complete lattice and $T\colon L\to L$ a monotonic mapping. Then, T has a least fixed point $lfp(T)$ and a greatest fixed point $gfp(T)$. Furthermore, it is verified that*

$$lfp(T) = \inf\{x \in L \mid T(x) = x\} = \inf\{x \in L \mid T(x) \preccurlyeq x\}$$
$$gfp(T) = \sup\{x \in L \mid T(x) = x\} = \sup\{x \in L \mid x \preccurlyeq T(x)\}$$

Proposition 2. *Let $\langle L, \preccurlyeq\rangle$ be a complete lattice and $T : L \to L$ a monotonic and continuous mapping. Then, $lfp(T) = T^{\omega}(\bot)$, where $T^{\omega}(\bot)$ denotes the iteration of T from the least element of the lattice $\bot$ and ω is the smallest limit ordinal, apart from $\bot$.*

In logic programming, the continuity is defined through the notion of directed set.

Definition 1. Let $\langle L, \preccurlyeq\rangle$ be a complete lattice. We say that $X \subseteq L$ is *directed* if each finite subset of X has an upper bound in X. Moreover, we say that T is *Scott-continuous* if it preserves suprema of directed sets, that is, given a directed set $X \subseteq L$ it is satisfied that

$$T(\sup X) = \sup\{T(x) \mid x \in X\}$$

A mapping $T\colon L^n \to L$ is said to be Scott-continuous if it is Scott-continuous in each argument separately.

From now on, we will refer to Scott-continuity simply as continuity. Next, we introduce a small summary of the most significant definitions of multi-adjoint logic programming. More details can be seen in [23, 24].

The algebraic structure considered in the multi-adjoint logic programming framework is given by the notion of multi-adjoint Ω-algebra. This definition is based on the notions of Ω-algebra, graded set, alphabet, and adjoint pair

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

[2, 23]. Namely, given an Ω-algebra composed of operational symbols, a multi-adjoint Ω-algebra is a tuple,[1] $\mathfrak{L} = (L, \preccurlyeq, \dot{\leftarrow}_1, \dot{\&}_1, \ldots, \dot{\leftarrow}_n, \dot{\&}_n)$, where $\langle L, \preccurlyeq\rangle$ is a bounded lattice, $(\dot{\leftarrow}_i, \dot{\&}_i)$ are adjoint pairs in $\langle L, \preccurlyeq\rangle$, satisfying $\top \dot{\&}_i \vartheta = \vartheta \dot{\&}_i \top = \vartheta$, for all $\vartheta \in L$ and $i \in \{1, \ldots, n\}$. Note that the tuple $\mathfrak{L}$ can contain other general aggregation operators associated with other symbols in the graded set Ω, different from $\leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n$, as we can see in the following example.

Example 1. Given the graded set $\Omega = \{\leftarrow_{\mathrm{P}}, \&_{\mathrm{P}}, \leftarrow_{\mathrm{G}}, \&_{\mathrm{G}}, \wedge_{Ł}, @_{(2,1)}\}$, the tuple $\mathfrak{L} = (L, \preccurlyeq, \dot{\leftarrow}_{\mathrm{P}}, \dot{\&}_{\mathrm{P}}, \dot{\leftarrow}_{\mathrm{G}}, \dot{\&}_{\mathrm{G}}, \dot{\wedge}_{Ł}, \dot{@}_{(2,1)})$ is a multi-adjoint Ω-algebra, where $\dot{\&}_{\mathrm{P}}$ and $\dot{\&}_{\mathrm{G}}$ are the product and Gödel t-norms respectively, together with their residuated implications, $\dot{\wedge}_{Ł}$ is the Łukasiewicz t-norm and $\dot{@}_{(2,1)}$ is an aggregation operator. These operators are defined for all $x, y, z \in [0, 1]$ as follows

$$\dot{\&}_{\mathrm{G}}(x, y) = \min\{x, y\} \qquad z \dot{\leftarrow}_{\mathrm{G}} y = \begin{cases} 1 & \text{if } y \leq z \\ z & \text{otherwise} \end{cases}$$

$$\dot{\&}_{\mathrm{P}}(x, y) = x \cdot y \qquad z \dot{\leftarrow}_{\mathrm{P}} y = \begin{cases} 1 & \text{if } y \leq z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$

$$\dot{\wedge}_{Ł}(x, y) = \max\{0, x + y - 1\} \qquad \dot{@}_{(2,1)}(x, y) = \frac{2x + y}{3}$$

Now, we recall the definition of multi-adjoint logic program.

Definition 2. Given a multi-adjoint Ω-algebra $\mathfrak{L}$ and a set of propositional symbols Π, a *multi-adjoint logic program* is a set $\mathbb{P}$ of weighted rules of the form $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$, also denoted $A \overset{\vartheta}{\longleftarrow}_i \mathcal{B}$, such that:

1. The head of the rule A is a propositional symbol of Π.
2. The body $\mathcal{B}$, which is a formula built from propositional symbols $B_1, \ldots, B_n$ ($n \geq 0$) by the use of conjunctions $\&_1, \ldots, \&_n$ and $\wedge_1, \ldots, \wedge_k$, disjunctions $\vee_1, \ldots, \vee_l$ and other more general $@_1, \ldots, @_m$, belong to $\mathfrak{L}$.
3. The confidence factor ϑ is an element (a truth-value) of the lattice L of $\mathfrak{L}$.

The particular case of a weighted rule where the body $\mathcal{B}$ is $\top \in L$ is called a *fact*. The following definition is given to simplify the notation.

Definition 3. Given a multi-adjoint Ω-algebra $\mathfrak{L}$ and any operator symbol ν in Ω, we say that ν is *continuous* if its interpretation, denoted as $\dot{\nu}$, is continuous in L.

Next, we show a result necessary to prove the continuity theorem.

Lemma 1. ([23]) *Let $\mathbb{P}$ be a program interpreted on a multi-adjoint Ω-algebra $\mathfrak{L}$, and let $\mathcal{B}$ be any body formula in $\mathbb{P}$. Assume that all the operators @ in $\mathcal{B}$ are continuous, let X be a directed set of interpretations, and write $S = \sup X$; then*

$$\hat{S}(\mathcal{B}) = \sup\{\hat{J}(\mathcal{B}) \mid J \in X\}$$

In this paper, the operator to be studied is the quantified immediate consequences operator $T_{\mathbb{P}}^{\mathrm{M}}$ introduced in [24], which generalizes the usual one through an arbitrary generalized quantifier [30].

Definition 4. Let $\mathbb{P}$ be a multi-adjoint logic program, where the set of rules with head $A \in \Pi$ is finite and it is denoted as $\mathbb{P}_A = \{u_1, \ldots, u_n\}$, and a family of fuzzy measures invariant with respect to the cardinality $\mathrm{M} = \{\mu_A : \mathcal{P}(\mathbb{P}_A) \to [0, 1] \mid A \in \Pi, \mathbb{P}_A \neq \emptyset\}$. We define *the M-quantified immediate consequences operator obtained from the family M* as the mapping $T_{\mathbb{P}}^{\mathrm{M}} : \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$ given for every $A \in \Pi$ and $I \in \mathcal{I}_{\mathfrak{L}}$ by

$$T_{\mathbb{P}}^{M}(I)(A) = \begin{cases} T_{\mathbb{P}}^{Q_{\mu_A}}(I)(A) & \text{if } \mathbb{P}_A \neq \emptyset \\ 0 & \text{if } \mathbb{P}_A = \emptyset \end{cases} \tag{1}$$

[1]We are denoting the interpretation of the operator symbols with a dot in the top of the symbol appearing in the graded set Ω.

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

where $T_{\mathbb{P}}^{Q_{\mu_A}}(I)(A) = Q_{\mu_A}\left(C_I^A\right) = \bigvee_{i=1}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu_A(\{u_1, \ldots, u_i\})$, the operator $\otimes$ verifies the following boundary conditions[2]: $x \otimes 1 = x, x \otimes 0 = 0 \otimes x = 0$, for all $x \in [0,1]$ and π is a permutation on $\{1, \ldots, n\}$ such that $C_I^A\left(u_{\pi(n)}\right) \leq \ldots \leq C_I^A\left(u_{\pi(1)}\right)$ and the mapping $C_I^A : \mathbb{P} \to [0,1]$, for each $\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}$, is defined as

$$C_I^A(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle) = \begin{cases} \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) & \text{if } C = A \\ 0 & \text{otherwise} \end{cases}$$

The monotonicity of $T_{\mathbb{P}}^{M}$ together with other properties was proved in [24]. The notion of model was also adapted to the new flexible framework arisen the definition of M-model [24]. Furthermore, the least M-model of the program can be obtained by iterating an ordinal number of times α the quantified immediate consequences operator starting from the least interpretation $\vartriangle$, as the following result ensures.

Theorem 1. ([24]) *Given a multi-adjoint logic program $\mathbb{P}$, we have that*

$$M_{\mathbb{P}} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is an } M\text{-model of } \mathbb{P}\}$$

is an M*-model and, for some ordinal α, it is verified that $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{M}\right) = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\vartriangle)$.*

3 | ON THE CONTINUITY OF THE $T_{\mathbb{P}}^{\mathrm{M}}$ OPERATOR

In this section we will study conditions to ensure the continuity of the new definition of the immediate consequences operator. As we have already mentioned, the continuity of the operator $T_{\mathbb{P}}^{\mathrm{M}}$ is essential in order to obtain its least fixed point in ω countable number of iterations. In the following, we will consider a multi-adjoint logic program $\mathbb{P}$ on the unit interval $([0,1], \leq)$, the adjoint pairs $(\leftarrow_i, \&_i)$ on $([0,1], \leq)$, with $i \in \{1, \ldots, n\}$, and the family of normalized fuzzy measures invariant with respect to the cardinality $\mathrm{M} = \{\mu_A : \mathcal{P}(\mathbb{P}_A) \to [0,1] \mid A \in \Pi\}$. The details of these definitions can be seen for example in [24].

Our first result is a generalization of the theorem in the classical fuzzy logic programming framework [23]. This theorem shows that if all operators considered in the program are continuous, then $T_{\mathbb{P}}^{\mathrm{M}}$ is continuous.

Theorem 2. *If all the operators occurring in the bodies of the rules of a program $\mathbb{P}$ are continuous, the adjoint conjunctors are continuous on the second argument and $\otimes$ is continuous on the first argument, then $T_{\mathbb{P}}^{\mathrm{M}}$ is continuous.*

Proof. Let $X \subseteq \mathcal{I}_{\mathfrak{L}}$ be a directed subset of interpretations, $A \in F_{\Omega}$ an atomic formula and $\langle A \leftarrow_i \mathcal{B}_i, \vartheta_i \rangle \in \mathbb{P}$, for all $i \in \{1, \ldots, n\}$, the set of rules with head A. We have to prove that

$$T_{\mathbb{P}}^{\mathrm{M}}(\sup X)(A) = \sup\left\{T_{\mathbb{P}}^{\mathrm{M}}(J)(A) \mid J \in X\right\} \tag{2}$$

Considering $S = \sup X$, the following chain of equalities is obtained.

$$\begin{aligned} T_{\mathbb{P}}^{\mathrm{M}}(\sup X)(A) &= T_{\mathbb{P}}^{Q_{\mu_A}}(\sup X)(A) \\ &= Q_{\mu_A}\left(C_S^A\right) \\ &= \bigvee_{i=1}^{n}\left(\vartheta_i \dot{\&}_i \hat{S}(\mathcal{B}_i)\right) \otimes \mu_A(\{u_1, \ldots, u_i\}) \\ &\overset{(1)}{=} \bigvee_{i=1}^{n}\left(\vartheta_i \dot{\&}_i \sup\left\{\hat{J}(\mathcal{B}_i) \mid J \in X\right\}\right) \otimes \mu_A(\{u_1, \ldots, u_i\}) \\ &\overset{(2)}{=} \bigvee_{i=1}^{n} \sup\left\{\vartheta_i \dot{\&}_i \hat{J}(\mathcal{B}_i) \mid J \in X\right\} \otimes \mu_A(\{u_1, \ldots, u_i\}) \end{aligned}$$

[2]These conditions are required to obtain the universal and existential quantifiers as particular cases.

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

$$\begin{aligned}
&\overset{(3)}{=} \bigvee_{i=1}^{n} \sup\left\{\left(\vartheta_i \dot{\&}_i \hat{J}(\mathcal{B}_i)\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \mid J \in X\right\} \\
&= \sup\left\{\bigvee_{i=1}^{n}\left(\vartheta_i \dot{\&}_i \hat{J}(\mathcal{B}_i)\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \mid J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{Q_{\mu_A}}(J)(A) \mid J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{\mathrm{M}}(J)(A) \mid J \in X\right\}
\end{aligned}$$

where equality (1) is satisfied due to Lemma 1, while equalities (2) and (3) follow from the continuity of the operator $\&_i$ on the second argument and $\otimes$ on the first one. Therefore, we can conclude that $T_{\mathbb{P}}^{\mathrm{M}}$ is continuous. □

In the following example, we will show that the consideration of directed sets cannot be dropped from the definition of continuity, in order to satisfy Equation (2).

Example 2. We consider a multi-adjoint Ω-algebra $\langle [0,1], \leq, \&_1, \&_2\rangle$, the program $\mathbb{P}$ composed of the rules $\langle A \leftarrow_1 B, \vartheta_1\rangle$ and $\langle A \leftarrow_2 B \wedge C, \vartheta_2\rangle$, with $0 \neq \vartheta_2 < \vartheta_1$, where $\&_1$,$\&_2$ can be interpreted as left-continuous t-norms and $\wedge$ as a t-norm [31]. Moreover, in this example, the family $M = \left\{\mu_A = \mu^1\right\}$, which is a singleton, is considered, where $\mu^1 : \mathcal{P}(\mathbb{P}) \to [0,1]$ is defined, for each $D \in \mathcal{P}(\mathbb{P})$, as

$$\mu^1(D) = \begin{cases} 0 \text{ if } D = \varnothing \text{ or } D = \{u_i\} \text{ for all } i \in \{1, \ldots, |\mathbb{P}|\} \\ 1 \text{ otherwise} \end{cases} \tag{3}$$

Let X be the set of interpretations $\{I_1, I_2\}$ given by

$$\begin{array}{lll} I_1(A) = 0 & I_1(B) = 1 & I_1(C) = 0 \\ I_2(A) = 0 & I_2(B) = 0 & I_2(C) = 1 \end{array}$$

Clearly X is not directed because $\{I_1, I_2\} \subseteq X$ has not an upper bound in X. Then, it is verified that

$$\begin{aligned}
\sup\left\{T_{\mathbb{P}}^{\mathrm{M}}(J)(A) \mid J \in X\right\} &= \sup\left\{T_{\mathbb{P}}^{Q_{\mu^1}}(J)(A) \mid J \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{Q_{\mu^1}}(I_1)(A), T_{\mathbb{P}}^{Q_{\mu^1}}(I_2)(A)\right\} \\
&= \sup\{0, 0\} = 0
\end{aligned}$$

On the other hand, $S = \sup X$ is defined as follows: $S(A) = 0, S(B) = 1, S(C) = 1$. Then, we obtain the following chain of equalities.

$$\begin{aligned}
T_{\mathbb{P}}^{\mathrm{M}}(\sup X)(A) &= T_{\mathbb{P}}^{Q_{\mu^1}}(\sup X)(A) \\
&= Q_{\mu^1}\left(C_S^A\right) \\
&= \bigvee_{i=1}^{2}\left(\vartheta_i \dot{\&}_i \hat{S}(\mathcal{B}_i)\right) \otimes \mu\left(\{u_1, \ldots, u_i\}\right) \\
&= \left((\vartheta_1 \dot{\&}_1 \hat{S}(B)) \otimes \mu\left(\{u_1\}\right)\right) \vee \left((\vartheta_2 \dot{\&}_2 \hat{S}(B \wedge C)) \otimes \mu\left(\{u_1, u_2\}\right)\right) \\
&= \left((\vartheta_1 \dot{\&}_1 1) \otimes 0\right) \vee \left((\vartheta_2 \dot{\&}_2 1) \otimes 1\right) = \vartheta_2
\end{aligned}$$

As $T_{\mathbb{P}}^{\mathrm{M}}(\sup X)(A) = \vartheta_2$, then Equality (2) does not hold for arbitrary subsets $X \subseteq \mathcal{I}_{\mathfrak{L}}$.

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

We present the following result in which we prove that the continuity of the adjoint conjuntor is a necessary condition for the continuity of the operator $T_{\mathbb{P}}^{\mathrm{M}}$.

Theorem 3. *If the operator $T_{\mathbb{P}}^{\mathrm{M}}$ is continuous for all program $\mathbb{P}$ in $\mathfrak{L}$, then the adjoint conjunctors are continuous on the second argument, $\otimes$ is continuous on the first argument, and any operator in the body of the rules is continuous.*

Proof. Given $\vartheta \in [0,1]$ and the directed set $Y \subseteq [0,1]$, we will prove first of all that the adjoint conjunctors $\dot{\&}_i$, with $i \in \{1, \dots, n\}$, are continuous on the second argument, that is,

$$\sup\left\{\vartheta \dot{\&}_i y \,|\, y \in Y\right\} = \vartheta \dot{\&}_i \sup Y$$

For this, we will consider the propositional symbols $A, B \in \Pi$ and the set of interpretations $X = \left\{I_y \,|\, y \in Y\right\}$ defined as

$$I_y(C) = \begin{cases} y & \text{if } C = B \\ 0 & \text{otherwise} \end{cases}$$

for all $C \in \Pi$. Now, we will show that X is a directed set. Given a subset $\left\{I_{y_1}, \dots, I_{y_m}\right\}$ of X, since $\{y_1, \dots, y_m\} \subseteq Y$ and Y is a directed set, then there exists $y_M \in Y$, such as it is an upper bound of $\{y_1, \dots, y_m\}$. Therefore, the interpretation I_{y_M} is an upper bound of $\left\{I_{y_1}, \dots, I_{y_m}\right\}$. Hence, X is directed. Next, consider the program $\mathbb{P}$ composed of the rule

$$u_1 = \langle A \leftarrow_i B,\ \vartheta \rangle$$

The following chain of equalities hold.

$$\begin{aligned}
\sup\left\{\vartheta \dot{\&}_i y \,|\, y \in Y\right\} &\overset{(1)}{=} \sup\left\{C_{I_y}^{A}(u_1) \otimes \mu_A(\{u_1\}) \,|\, I_y \in X\right\} \\
&\overset{(2)}{=} \sup\left\{Q_{\mu_A}\left(C_{I_y}^{A}\right) \,|\, I_y \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{Q_{\mu_A}}\left(I_y\right)(A) \,|\, I_y \in X\right\} \\
&= \sup\left\{T_{\mathbb{P}}^{\mathrm{M}}\left(I_y\right)(A) \,|\, I_y \in X\right\} \\
&\overset{(3)}{=} T_{\mathbb{P}}^{\mathrm{M}}(\sup X)(A) \\
&= T_{\mathbb{P}}^{Q_{\mu_A}}(\sup X)(A) \\
&\overset{(2)}{=} C_{\sup X}^{A}(u_1) \otimes \mu_A(\{u_1\}) \\
&\overset{(1)}{=} \vartheta \dot{\&}_i \sup Y
\end{aligned}$$

where (1) follows from the definition of $I_y(C)$ and because of μ is a normalized measure, and so $\mu_A(\{u_1\}) = 1$. Moreover, (2) holds because of only one rule with head A exists in the program, and (3) is verified due to the continuity of $T_{\mathbb{P}}^{\mathrm{M}}$. Therefore, we conclude that $\sup\left\{\vartheta \dot{\&}_i y \,|\, y \in Y\right\} = \vartheta \dot{\&}_i \sup Y$ and $\&_i$ is continuous on the second argument, for all $i \in \{1, \dots, n\}$.

The continuity on the first argument of $\otimes$ can be proved considering the same interpretation I_y, the program formed with the rules $u_1 = \langle A \leftarrow_i B, \top \rangle$ and $u_2 = \langle A \leftarrow_i B', \top \rangle$, and the fuzzy measure defined for each $D \subseteq \mathcal{P}(\mathbb{P})$ as

$$\mu^2(D) = \begin{cases} 0 \text{ if } D = \varnothing \\ x \text{ if } D = \{u_1\} \text{ or } D = \{u_2\} \\ 1 \text{ otherwise} \end{cases}$$

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

with $x \in [0,1]$. We obtain the following chain of equalities.

$$\begin{aligned}
\sup\{y \otimes x | y \in Y\} &= \sup\{\sup\{(\top \dot{\&}_i y) \otimes x, (\top \dot{\&}_i 0) \otimes 1\} | y \in Y\} \\
&= \sup\{\sup\{(\top \dot{\&}_i I_y(B)) \otimes \mu^2(\{u_1\}), (\top \dot{\&}_i I_y(B')) \otimes \mu^2(\{u_1, u_2\})\} | y \in Y\} \\
&= \sup\left\{ T_{\mathbb{P}}^{Q_{\mu^2}}\left(I_y\right)(A) | I_y \in X \right\} \\
&\overset{(*)}{=} T_{\mathbb{P}}^{Q_{\mu^2}}(\sup X)(A) \\
&= \sup\{C^A_{\sup X}(u_1) \otimes \mu^2(\{u_1\}), C^A_{\sup X}(u_2) \otimes \mu^2(\{u_1, u_2\})\} \\
&= \sup\{(\top \dot{\&}_i \sup X(B)) \otimes \mu^2(\{u_1\}), (\top \dot{\&}_i \sup X(B')) \otimes \mu^2(\{u_1, u_2\})\}\} \\
&= (\top \dot{\&}_i \sup X(B)) \otimes \mu^2(\{u_1\}) \\
&= \sup Y \otimes x
\end{aligned}$$

where $(*)$ holds from the definition of $T_{\mathbb{P}}^{\mathrm{M}}$ and its continuity.

Next, we continue with the study of the continuity of the operators in the body of the rules. In order to simplify the process we will use the notation $\bar{y} \in [0,1]^n$ for $\bar{y} = (y_1, \ldots, y_n)$, where $y_1, \ldots, y_n \in [0,1]$ and, given $\bar{y}, \bar{z} \in [0,1]^n$, we say that $\bar{y} \leq \bar{z}$, if $y_i \leq z_i$, for all $i \in \{1, \ldots, n\}$. Given a directed set $Y \subseteq [0,1]^n$ and an operator symbol @ in the body of one rules in $\mathbb{P}$, we have to prove that

$$\dot{@}(\sup Y) = \sup\left\{ \dot{@}(y_1, \ldots, y_n) \mid (y_1, \ldots, y_n) \in Y \right\}$$

Clearly, if $(y_1, \ldots, y_n) \in Y$, then $(y_1, \ldots, y_n) \leq \sup Y$ and by the monotonicity of $\dot{@}$, it is verified that $\dot{@}(y_1, \ldots, y_n) \leq \dot{@}(\sup Y)$. Therefore,

$$\sup\left\{ \dot{@}(y_1, \ldots, y_n) \mid (y_1, \ldots, y_n) \in Y \right\} \leq \dot{@}(\sup Y) \tag{4}$$

In order to prove the other inequality, given the propositional symbols $A_1, \ldots, A_n \in \Pi$ and $\bar{y} = (y_1, \ldots, y_n) \in Y$, we can consider the interpretation $I_{\bar{y}}$ defined as

$$I_{\bar{y}}(C) = \begin{cases} y_i & \text{if } C = A_i \\ 0 & \text{otherwise} \end{cases}$$

On the one hand, consider the set $X_Y = \left\{ I_{\bar{y}} \in \mathcal{I}_{\mathfrak{L}} \mid \bar{y} \in Y \right\}$ and its supremum $S_Y = \sup X_Y$. Given that Y is directed, from the definition of $I_{\bar{y}}$, X_Y is also directed, and therefore $S_Y \in X_Y$. For all $\bar{y} \in Y$ it is verified that $\bar{y} = (y_1, \ldots, y_n) = \left(I_{\bar{y}}(A_1), \ldots, I_{\bar{y}}(A_n)\right) \leq (S_Y(A_1), \ldots, S_Y(A_n))$ and consequently, $\sup Y \leq (S_Y(A_1), \ldots, S_Y(A_n))$. As a consequence, from the monotonicity of $\dot{@}$ and the homomorphic extension theorem, we reach that

$$\dot{@}(\sup Y) \leq \dot{@}(S_Y(A_1), \ldots, S_Y(A_n)) = \widehat{S_Y}(@(A_1, \ldots, A_n))$$

On the other hand, consider the multi-adjoint logic program $\mathbb{P}$ composed of the rule

$$u_1 = \langle A \leftarrow_i @(A_1, \ldots, A_n), \top \rangle$$

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

The following chain of equalities holds.

$$\begin{aligned}
\widehat{S_Y}\left(@(A_1,\ldots,A_n)\right) &\overset{(1)}{=} \top \dot{\&}_i \widehat{S_Y}\left(@\left(A_1,\ldots,A_n\right)\right)\\
&\overset{(2)}{=} C^{\mathcal{A}}_{S_Y}(u_1) \otimes \mu_A(\{u_1\})\\
&= T^{Q_{\mu_A}}_{\mathbb{P}}(S_Y)(A)\\
&= T^{\mathrm{M}}_{\mathbb{P}}(S_Y)(A)\\
&\overset{(3)}{=} \sup\left\{T^{\mathrm{M}}_{\mathbb{P}}\left(I_{\bar{y}}\right)(A) \,|\, I_{\bar{y}} \in X_Y\right\}\\
&= \sup\left\{T^{Q_{\mu_A}}_{\mathbb{P}}\left(I_{\bar{y}}\right)(A) \,|\, I_{\bar{y}} \in X_Y\right\}\\
&= \sup\left\{C^{\mathcal{A}}_{I_{\bar{y}}}(u_1) \otimes \mu_A(\{u_1\}) \,|\, I_{\bar{y}} \in X_Y\right\}\\
&= \sup\left\{\top \dot{\&}_i \widehat{I_{\bar{y}}}\left(@\left(A_1,\ldots,A_n\right)\right) \otimes \mu_A(\{u_1\}) \,|\, I_{\bar{y}} \in X_Y\right\}\\
&= \sup\left\{\dot{@}\left(y_1,\ldots,y_n\right) \mid \left(y_1,\ldots,y_n\right) \in Y\right\}
\end{aligned}$$

where (1) is due to the boundary conditions of the operator $\dot{\&}_i$, (2) because of μ is a normalized measure and the definition of $C^{\mathcal{A}}_{S_Y}$, and (3) follows from the continuity of $T^{\mathrm{M}}_{\mathbb{P}}$. Consequently, and also from Expression (4), we have reached that

$$\begin{aligned}
\sup\left\{\dot{@}\left(y_1,\ldots,y_n\right) \,|\, \bar{y} \in Y\right\} &\leq \dot{@}\left(\sup Y\right)\\
&\leq \widehat{S_Y}\left(@\left(A_1,\ldots,A_n\right)\right)\\
&= \sup\left\{\dot{@}\left(y_1,\ldots,y_n\right) \mid \left(y_1,\ldots,y_n\right) \in Y\right\}
\end{aligned}$$

□

Finally, we obtain that the least M-model of a program $\mathbb{P}$ can be reached computationally.

Theorem 4. *If $T^{\mathrm{M}}_{\mathbb{P}}$ is continuous, then $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\omega}(\vartriangle)$ is the least model of $\mathbb{P}$, that is, $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\omega}(\vartriangle) = M_{\mathbb{P}}$.*

Proof. The proof follows from Proposition 2 and Theorem 1. □

Finally, the following example will illustrate how the least model of a program is obtained and how the continuity influences the process.

Example 3. In this example, we have designed a logic program in order to determine whether a patient has the covid disease or not. For this, in order to construct the rules of the program, we consider the multi-adjoint Ω-algebra $\left\langle [0,1], \leq, \dot{\leftarrow}_{\mathrm{G}}, \dot{\&}_{\mathrm{G}}, \dot{\leftarrow}_{\mathrm{P}}, \dot{\&}_{\mathrm{P}}, \dot{\wedge}_{Ł}, \dot{@}_{(2,1)} \right\rangle$, where $\dot{\&}_{\mathrm{G}}$ and $\dot{\&}_{\mathrm{P}}$ are the Gödel and the product conjunctors, respectively, with $\dot{\leftarrow}_{\mathrm{G}}$ and $\dot{\leftarrow}_{\mathrm{P}}$ being their respective adjoint implications. Moreover, we use the extra operators $\dot{\wedge}_{Ł}$, the Łukasiewicz conjunction and the aggregator $\dot{@}_{(2,1)}$.

In addition, we will consider the set of propositional symbols

$$\Pi = \{\texttt{covid}, \texttt{diarrhoea}, \texttt{headache}, \texttt{moderate fever}, \texttt{cough}, \texttt{waist pain}\}$$

In an specific situation, after analyzing it with experts, we can consider the following multi-adjoint logic program $\mathbb{P}$ for determining the covid disease in a patient.

$$\begin{aligned}
&\texttt{covid} \overset{0.8}{\leftarrow}_{\mathrm{G}} @_{(2,1)}(\texttt{diarrhoea} \wedge_{Ł} \texttt{headache}, \texttt{waist pain})\\
&\texttt{covid} \overset{0.9}{\leftarrow}_{\mathrm{P}} \texttt{moderate fever}\\
&\texttt{covid} \overset{0.7}{\leftarrow}_{\mathrm{P}} \texttt{cough}
\end{aligned}$$

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

The program is finished with the values provided with each particular patient, in the form of facts. For example, a person may have the symptoms represented by the following facts.

$$\texttt{moderate fever} \xleftarrow{0.5} 1 \qquad \texttt{headache} \xleftarrow{0.7} 1$$
$$\texttt{cough} \xleftarrow{0.8} 1 \qquad \texttt{waist pain} \xleftarrow{0.3} 1$$
$$\texttt{diarrhoea} \xleftarrow{0.6} 1$$

We will consider the family of fuzzy measures defined as

$$M = \left\{\mu_{\texttt{covid}} = \mu^1, \mu_A = \mu^\exists \text{ for all } A \in \Pi \backslash \{\texttt{covid}\}\right\}$$

with the fuzzy measure μ^1 given in Expression (3), in order to consider the second greatest value due to the expert estimates that there may be a lot of noise in the data.[3] We will show how the least M-model of the program is obtained iterating the $T_{\mathbb{P}}^{\mathrm{M}}$ operator from the least interpretation $\vartriangle$, taking into account this family M. In order to simplify, we will denote the propositional symbols as $\Pi = \{\texttt{covid}, \texttt{d}, \texttt{h}, \texttt{mf}, \texttt{c}, \texttt{wp}\}$. For example, for the propositional symbol $\texttt{covid}$, the first iteration is the following.

$$\begin{aligned}
T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)(\texttt{covid}) &= \bigvee_{i=1}^{3} C_{\vartriangle}^{\texttt{covid}}(u_{\pi(i)}) \otimes \mu^1(\{u_1, \ldots, u_i\}) \\
&= \sup\left\{\left(0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)}\left(\vartriangle(\texttt{d}) \dot{\lambda}_{\mathrm{Ł}} \vartriangle(\texttt{h}), \vartriangle(\texttt{wp})\right)\right) \otimes \mu^1(\{u_1\}),\right. \\
&\quad \left.\left(0.9 \dot{\&}_{\mathrm{P}} \vartriangle(\texttt{mf})\right) \otimes \mu^1(\{u_1, u_2\}), \left(0.7 \dot{\&}_{\mathrm{P}} \vartriangle(\texttt{c})\right) \otimes \mu^1(\{u_1, u_2, u_3\})\right\} \\
&= \sup\left\{\left(0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)}\left(0 \dot{\lambda}_{\mathrm{Ł}} 0, 0\right)\right) \otimes 0, \left(0.9 \dot{\&}_{\mathrm{P}} 0\right) \otimes 1, \left(0.7 \dot{\&}_{\mathrm{P}} 0\right) \otimes 1\right\} \\
&= \sup\{0 \otimes 0, 0 \otimes 1, 0 \otimes 1\} \\
&= \sup\{0, 0, 0\} = 0
\end{aligned}$$

In the following table we show the values obtained for each propositional symbol and iteration (Table 1).

For example, the second iteration for covid is

$$\begin{aligned}
\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^2(\vartriangle)(\texttt{covid}) &= \bigvee_{i=1}^{3} C_{T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)}^{\texttt{covid}}(u_{\pi(i)}) \otimes \mu^1(\{u_1, \ldots, u_i\}) \\
&= \sup\left\{\left(0.7 \dot{\&}_{\mathrm{P}} T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)(\texttt{c})\right) \otimes \mu^1(\{u_1\}), \left(0.9 \dot{\&}_{\mathrm{P}} T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)(\texttt{mf})\right) \otimes \mu^1(\{u_1, u_2\}),\right. \\
&\quad \left.\left(0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)}\left(T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)(\texttt{d}) \dot{\lambda}_{\mathrm{Ł}} T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)(\texttt{h}), T_{\mathbb{P}}^{\mathrm{M}}(\vartriangle)(\texttt{wp})\right)\right) \otimes \mu^1(\{u_1, u_2, u_3\})\right\} \\
&= \sup\left\{\left(0.7 \dot{\&}_{\mathrm{P}} 0.8\right) \otimes 0, \left(0.9 \dot{\&}_{\mathrm{P}} 0.5\right) \otimes 1, \left(0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)}\left(0.6 \dot{\lambda}_{\mathrm{Ł}} 0.7, 0.3\right)\right) \otimes 1\right\} \\
&= \sup\{0.3 \otimes 1, 0.45 \otimes 1, 0.56 \otimes 0\} \\
&= \sup\{0.3, 0.45, 0\} = 0.45
\end{aligned}$$

Notice that the values for the propositional symbols are not zero in this iteration and the second greatest value (as the expert decides) given by the rules has been considered. Therefore, the least fixed point of the quantified immediate consequences operator, which is the least M-model of the program, corresponds to the second iteration of the operator, that is, $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^2(\vartriangle)$.

[3]Note that this kind of strategies are really effective when more rules and data are present in the data set.

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 Iterations of the quantified immediate consequences operator until the least fixed point is obtained.

	Δ	$T^{\mathbf{M}}_{\mathbb{P}}(\Delta)$	$\left(T^{\mathbf{M}}_{\mathbb{P}}\right)^2(\Delta)$	$\left(T^{\mathbf{M}}_{\mathbb{P}}\right)^3(\Delta)$
`moderate fever`	0	0.5	0.5	0.5
`cough`	0	0.8	0.8	0.8
`diarrhoea`	0	0.6	0.6	0.6
`headache`	0	0.7	0.7	0.7
`waist pain`	0	0.3	0.3	0.3
`covid`	0	0	0.45	0.45

Now, consider the adjoint pair $(\dot{\&}_{\mathrm{d}}, \dot{\leftarrow}_{\mathrm{d}})$, where the t-norm $\dot{\&}_{\mathrm{d}} : [0,1]\times[0,1] \to [0,1]$ and the residuated implication $\dot{\leftarrow}_{\mathrm{d}} : [0,1]\times[0,1] \to [0,1]$ are defined as

$$x\dot{\&}_{\mathrm{d}}y = \begin{cases} \min\{x,y\} & \text{if } 0.5 \le x, y \\ 0 & \text{otherwise} \end{cases} \qquad z\dot{\leftarrow}_{\mathrm{d}}\, y = \begin{cases} 1 \text{ if } y \le z \text{ and } 0.5 \le x, y \\ z \text{ if } y > z \text{ and } 0.5 \le x, y \\ 1 \text{ otherwise} \end{cases}$$

and we will change the algebraic structure used in the previous logic program from $\langle [0,1], \le, \dot{\leftarrow}_{\mathrm{G}}, \dot{\&}_{\mathrm{G}}, \dot{\leftarrow}_{\mathrm{P}}, \dot{\&}_{\mathrm{P}}, \dot{\wedge}_{\mathrm{Ł}}, \dot{@}_{(2,1)} \rangle$ to the multi-adjoint Ω-algebra $\langle [0,1], \le, \dot{\leftarrow}_{\mathrm{d}}, \dot{\&}_{\mathrm{d}}, \dot{\wedge}_{\mathrm{Ł}}, \dot{@}_{(2,1)} \rangle$, that is, the logic rules from the program are

$$\texttt{covid} \overset{0.8}{\leftarrow}_{\mathrm{d}} @_{(2,1)}(\texttt{diarrhoea} \wedge_{\mathrm{Ł}} \texttt{headache}, \texttt{waist pain})$$
$$\texttt{covid} \overset{0.9}{\leftarrow}_{\mathrm{d}} \texttt{moderate fever}$$
$$\texttt{covid} \overset{0.7}{\leftarrow}_{\mathrm{d}} \texttt{cough}$$

In this case, we will see that if we are less sure about the veracity of a given fact, that is, the value of the confidence factor of one of the facts changes slightly, then the veracity about the conclusion will change drastically.

$$\begin{aligned}
\left(T^{\mathbf{M}}_{\mathbb{P}}\right)^2(\Delta)(\texttt{covid}) &= \bigvee_{i=1}^{3} C^{\texttt{covid}}_{T^{\mathbf{M}}_{\mathbb{P}}(\Delta)}(u_{\pi(i)}) \otimes \mu^1(\{u_1, \ldots, u_i\}) \\
&= \sup\Big\{\big(0.8\dot{\&}_{\mathrm{d}}\dot{@}_{(2,1)}\left(T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{d})\dot{\wedge}_{\mathrm{Ł}}T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{h}), T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{wp})\right)\big) \otimes \mu^1(\{u_1\}), \\
&\quad \big(0.9\dot{\&}_{\mathrm{d}}T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{mf})\big) \otimes \mu^1(\{u_1,u_2\}), \big(0.7\dot{\&}_{\mathrm{d}}T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{c})\big) \otimes \mu^1(\{u_1,u_2,u_3\})\Big\} \\
&= \sup\Big\{\big(0.8\dot{\&}_{\mathrm{d}}\dot{@}_{(2,1)}\left(0.6\dot{\wedge}_{\mathrm{Ł}}0.7, 0.3\right)\big) \otimes 1, \big(0.9\dot{\&}_{\mathrm{d}}0.5\big) \otimes 1, \big(0.7\dot{\&}_{\mathrm{d}}0.8\big) \otimes 0\Big\} \\
&= \sup\{0 \otimes 1, 0.5 \otimes 1, 0.7 \otimes 0\} \\
&= \sup\{0, 0.5, 0\} = 0.5
\end{aligned}$$

Now, the value of the fact associated with the propositional symbol `moderate fever` will be changed from 0.5 to 0.49. Then,

$$\begin{aligned}
\left(T^{\mathbf{M}}_{\mathbb{P}}\right)^2(\Delta)(\texttt{covid}) &= \bigvee_{i=1}^{3} C^{\texttt{covid}}_{T^{\mathbf{M}}_{\mathbb{P}}(\Delta)}(u_{\pi(i)}) \otimes \mu^1(\{u_1, \ldots, u_i\}) \\
&= \sup\Big\{\big(0.8\dot{\&}_{\mathrm{d}}\dot{@}_{(2,1)}\left(T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{d})\dot{\wedge}_{\mathrm{Ł}}T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{h}), T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{wp})\right)\big) \otimes \mu^1(\{u_1\}), \\
&\quad \big(0.9\dot{\&}_{\mathrm{d}}T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{mf})\big) \otimes \mu^1(\{u_1,u_2\}), \big(0.7\dot{\&}_{\mathrm{d}}T^{\mathbf{M}}_{\mathbb{P}}(\Delta)(\mathrm{c})\big) \otimes \mu^1(\{u_1,u_2,u_3\})\Big\} \\
&= \sup\Big\{\big(0.8\dot{\&}_{\mathrm{d}}\dot{@}_{(2,1)}\left(0.6\dot{\wedge}_{\mathrm{Ł}}0.7, 0.3\right)\big) \otimes 1, \big(0.9\dot{\&}_{\mathrm{d}}0.49\big) \otimes 1, \big(0.7\dot{\&}_{\mathrm{d}}0.8\big) \otimes 0\Big\} \\
&= \sup\{0 \otimes 1, 0 \otimes 1, 0.7 \otimes 0\} \\
&= \sup\{0, 0, 0\} = 0
\end{aligned}$$

Consequently, if the value of the fact changes slightly, from 0.5 to 0.49, the value of the consequences changes drastically from 0.5 to 0, underlining the non-commutative character of $T^{\mathbf{M}}_{\mathbb{P}}$.

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

4 | CONCLUSIONS AND FUTURE WORK

Necessary and sufficient conditions have been introduced in order to ensure the continuity of the new flexible quantified immediate consequences operator [24]. These results extend the ones given in [23], despite the new operator complicates its definition by considering generalized quantifiers in order to provide a more flexible operator focused on obtaining better models of real data sets. As a consequence, the least model (least fixed-point) of the operator $T_{\mathbb{P}}^{\mathrm{M}}$ can be reached in a countable number of iterations, providing the possibility of obtaining this fundamental model through operational semantics.

In the future, more properties will be studied. A relevant challenge will also be to extend the definition of $T_{\mathbb{P}}^{\mathrm{M}}$ on general lattices. Once the continuity can easily be checked, it would be significant to analyze the design of an optimal operational semantics. In addition, the obtained results will be implemented to complement the developments given, for instance, in Bousi-Prolog [11, 12] and FASILL [10]. Real-world scenarios will also be considered in the future.

AUTHOR CONTRIBUTIONS

Jesús Medina: Investigation; Validation; Writting; Methodoloy. **José Antonio Torné-Zambrano:** Investigation; Validation; Writting; Methodology.

ACKNOWLEDGEMENTS

The author would like to thank the anonymous referees for their careful reading of the paper and many useful suggestions to clarify this work.

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflicts of interest regarding the publication of this paper.

ORCID

Jesús Medina https://orcid.org/0000-0002-3931-5873

REFERENCES

1. J. A. Robinson, *A machine-oriented logic based on the resolution principle*, J. Assoc. Comput. Mach. (ACM) **12** (1965), no. 1, 23–41.
2. J. Lloyd, *Foundations of logic programming*, Springer Verlag, 1987.
3. O. Abu Arqub, J. Singh, B. Maayah, and M. Alhodaly, *Reproducing kernel approach for numerical solutions of fuzzy fractional initial value problems under the Mittag-Leffler kernel differential operator*, Math. Meth. Appl. Sci. **46** (2023), no. 7, 7965–7986.
4. J. A. Aledo, G. Manjabacas, J. Robles, and J. C. Valverde, *Fuzzy parallel dynamical systems on Zadeh operators*, Math. Meth. Appl. Sci. **46** (2023), no. 9, 10260–10267.
5. C. V. Damásio and L. M. Pereira, Monotonic and residuated logic programs, *In Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, Lecture Notes in Artificial Intelligence, Vol. **2143**, Springer, Berlin Heidelberg, 2001, pp. 748–759.
6. D. Lobo, V. López-Marchante, and J. Medina, *On the impact of sup-compositions in the resolution of multi-adjoint relation equations*, Math. Meth. Appl. Sci. **46** (2023), 15581–15598.
7. N. Madrid and M. Ojeda-Aciego, *A measure of consistency for fuzzy logic theories*, Math. Meth. Appl. Sci. **46** (2023), no. 15, 15982–15995.
8. G. C. Milanese and G. Pasi, *Similarity-based reasoning with order-sorted feature logic*, IEEE Trans. Fuzzy Syst. **32** (2024), no. 5, 2797–2810.
9. D. Mohapatra and S. Chakraverty, *Type-2 fuzzy linear system of equations with application in static problem of structures*, Math. Meth. Appl. Sci. **46** (2023), no. 1, 840–866.
10. P. Julián-Iranzo, G. Moreno, and J. A. Riaza, *The fuzzy logic programming language FASILL: design and implementation*, Int. J. Approx. Reason. **125** (2020), 139–168.
11. P. Julián-Iranzo and C. Rubio-Manzano, *Bousi-Prolog—a fuzzy logic programming language for modeling vague knowledge and approximate reasoning*, ICFC-ICNC 2010 - Proceedings of the International Conference on Fuzzy Computation and International Conference on Neural Computation, [parts of the International Joint Conference on Computational Intelligence IJCCI, Valencia, Spain *Edited by* J. Filipe and J. Kacprzyk. SciTePress, 2010, pp. 2010.
12. P. Julián-Iranzo and F. Sáenz-Pérez, *Bousi-Prolog: design and implementation of a proximity-based fuzzy logic programming language*, Expert Syst. Appl. **213** (2023), 118858.
13. L. Tomás, C. Vázquez Pérez-Íñigo, J. Tordsson, and G. Moreno, *Reducing noisy-neighbor impact with a fuzzy affinity-aware scheduler*, In 2015 International Conference on Cloud and Autonomic Computing, 2015, pp. 33–44.

RIGHTSLINK

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

14. J. M. Almendros-Jiménez, A. Becerra-Terón, and G. Moreno, *Fuzzy queries of social networks with FSA-SPARQL*, Expert Syst. Appl. **113** (2018), 128–146.
15. J. M. Almendros-Jiménez, M. Bofill, A. Luna-Tedesqui, G. Moreno, C. Vázquez, and M. Villaret, Fuzzy XPath for the automatic search of fuzzy formulae models, *Scalable Uncertainty Management*, Springer International Publishing, Cham, 2015, pp. 385–398.
16. G. Moreno, J. Pérez, and J. A. Riaza, Fuzzy logic programming for tuning neural networks, *Rules and Reasoning*, Springer International Publishing, Cham, 2019, pp. 190–197.
17. A. Srinivasan, A. Baskar, T. Dash, and D. Shah, *Composition of relational features with an application to explaining black-box predictors*, Mach. Learn. **113** (2024), no. 3, 1091–1132.
18. M. Mahout, R. P. Carlson, L. Simon, and S. Peres, *Logic programming-based minimal cut sets reveal consortium-level therapeutic targets for chronic wound infections*, NPJ Syst. Biol. Appl. **10** (2024), no. 1, 34.
19. F. P. Romero, P. Julián-Iranzo, A. Soto, M. Ferreira-Satler, and J. Gallardo-Casero, *Classifying unlabeled short texts using a fuzzy declarative approach*, Lang. Resour. Eval. **47** (2013), no. 1, 151–178.
20. C. Rubio-Manzano and P. J. Iranzo, *Incorporation of abstraction capability in a logic-based framework by using proximity relations*, J. Intell. Fuzzy Syst. **29** (2015), 1671–1683.
21. E. Çakir and Z. Ulukan, *A fuzzy linguistic programming for sustainable ecotourism activities*, In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC) 01, 2020.
22. J. Almendros-Jimenez, A. Becerra-Terón, G. Moreno, and J. Riaza, *Tuning fuzzy SPARQL queries in a fuzzy logic programming environment*, In 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2019, pp. 1–7.
23. J. Medina, M. Ojeda-Aciego, and P. Vojtáš, *Multi-adjoint logic programming with continuous semantics*, Lecture Notes Artif. Intell. **2173** (2001), 351–364.
24. J. Medina and J. A. Torné-Zambrano, *Immediate consequences operator on generalized quantifiers*, Fuzzy Set. Syst. **456** (2022), 72–91.
25. N. Cao, M. Holčapek, and M. Štěpnička, *Extensions of fuzzy relational compositions based on generalized quantifiers*, Fuzzy Sets Syst. **339** (2018), 73–98. Theme: Algebra and Set Theory.
26. V. Novák, *A formal theory of intermediate quantifiers*, Fuzzy Sets Syst. **159** (2008), no. 10, 1229–1246. Mathematical and Logical Foundations of Soft Computing.
27. J. Medina, M. Ojeda-Aciego, and P. Vojtáš, *Similarity-based unification: a multi-adjoint approach*, Fuzzy Set. Syst. **146** (2004), 43–62.
28. C. V. Damásio and L. M. Pereira, Hybrid probabilistic logic programs as residuated logic programs, *In Logics in Artificial Intelligence, JELIA'00*, Vol. **1919**, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 57–73.
29. A. Tarski, *A lattice-theoretical fixpoint theorem and its applications*, Pacific J. Math. **5** (1955), no. 2, 285–309.
30. M. Stěpnička and M. Holčapek, Fuzzy relational compositions based on generalized quantifiers, *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Springer International Publishing, Cham, 2014, pp. 224–233.
31. E. Klement, R. Mesiar, and E. Pap, *Triangular norms*, Kluwer academic, 2000.

How to cite this article: J. Medina and J. Antonio Torné-Zambrano, *Computational reaching of quantified consequences from imperfect initial data*, Math. Meth. Appl. Sci. (2024), 1–12, DOI 10.1002/mma.10542.

10991476, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/mma.10542 by Readcube (Labtiva Inc.), Wiley Online Library on [05/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

RIGHTSLINK

Capítulo 5

Técnica computacional para la obtención de consecuencias en programación lógica multiadjunta con cuantificadores generalizados

Título:	Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers
Autores:	Jesús Medina, José Antonio Torné-Zambrano
Revista:	Applied and Computational Mathematics
Factor de impacto JCR:	4,6. Posición 2 de 332 (Q1)
Año:	2024
Categoría:	Mathematics, Applied
Publicación:	Noviembre 2024
DOI:	https://doi.org/10.22111/IJFS.2023.7696

Appl. Comput. Math., V.23, N.4, 2024, pp.xx-xx
DOI: 10.30546/1683-6154.23.4.2024

COMPUTATIONAL TECHNIQUE FOR OBTAINING CONSEQUENCES IN MULTI-ADJOINT LOGIC PROGRAMMING WITH GENERALIZED QUANTIFIERS

J. MEDINA[1], J.A. TORNÉ-ZAMBRANO[1,*]

Abstract. One of the most widely used computational technique in logic programming is operational semantics, which allows the implementation of an automatic reasoning procedure for obtaining information and consequences from a knowledge system modelled by means of logical rules. This paper will introduce an operational semantics for the flexible framework of propositional multi-adjoint logic programming with generalized quantifiers. This approach extends diverse general logic programming frameworks, weakening the existential feature of the supremum operator and considering a malleable algebraic structure.

Keywords: Immediate Consequences Operator, Procedural Semantics, Fixed-Point Semantics, Fuzzy Sets, Generalized Quantifiers.

AMS Subject Classification:

1. Introduction

Artificial Intelligence is a fundamental discipline nowadays. Many tools in the fields of mathematics, computer science, and transversal ones, have been developed and applied to a huge range of real problems [4, 5, 32, 35]. However, a lot of challenges remain to be solved. A relevant research area which helps to achieve these challenges is mathematical logic and its variants considering fuzzy sets [13, 20, 37].

One of these variants is fuzzy logic programming [16, 18, 21, 23, 36], which is focused on modelling (real) problems through the use of logic rules. Fuzzy logic programming has been used in different areas, such as in supply-chain management [1], ontology [17], and environmental automatic control [1]. Specifically, it allows an executable language and so, it can be used to design algorithms and procedures for complementing the design of decision support systems [2,8,22,25].

In this research line, the definition of an operational semantics associated with a logic allows the implementation of an automatic reasoning procedure for obtaining information and deductions to be considered in the designed decision support system.

This paper will introduce an operational semantics for the adaptable framework of propositional multi-adjoint logic programming with generalized quantifiers [30, 31]. This approach extends diverse general logic programming frameworks [10, 21, 28, 36] weakening the existential feature of the supremum operator and considering a flexible algebraic structure [7]. These properties are relevant because the semantics of a logic program is characterized through the post-fixed points of the immediate consequences operator $T_{\mathbb{P}}$, which is defined from the supremum of a set of values. The strict nature of the supremum can cause that the presence of

Partially supported by the 2014–2020 ERDF Operational Programme in collaboration with the State Research Agency (AEI) in projects PID2019-108991GB-I00 and PID2022-137620NB-I00, and with the Ecological and Digital Transition Projects 2021 of the Ministry of Science and Innovation in project TED2021-129748B-I00.

[1]Department of Mathematics, University of Cádiz, Cádiz, Spain, e-mail: joseantonio.torne@uca.es
[*]Corresponding author e-mail: jesus.medina@uca.es
Manuscript received 14 December 2023.

noise in the data alters the final result of the computation. For this reason, a more general definition of the immediate consequences operator $T_{\mathbb{P}}^{\mathrm{M}}$, defined on the multi-adjoint framework and based on generalized quantifiers, was proposed in [31]. The multi-adjoint approach presents several advantages that allow the representation of more complex systems than with the classical logic programming (monotone and residuated) [10] and the fuzzy approach given in [36], among others. The use of several adjoint pairs in the construction of logic programs is allowed, as well as less restrictive properties in the considered connectives, such as the requirement of the commutativity and associativity properties. Therefore, this generalization provides a more flexible and versatile operator with interesting properties, which have been considered in different frameworks, such as in formal concept analysis [6].

Specifically, this paper will adapt the notions of computed and correct answers, and the procedure for obtaining computed answers to the flexible framework given by the multi-adjoint algebras and generalized quantifiers. In addition, a novel definition of reductant will be introduced in order to prove the soundness of the introduced procedure. Reductants were originally introduced in generalized annotated logic programming [19], and latter was considered in the multi-adjoint logic programming framework in [14, 26]. The notion of reductant will be fundamental in this paper to analyze the completeness, with the goal of obtaining an equivalence in terms of the corresponding maximum computed and correct answers.

The structure of the paper is as follows: in Section 2 some preliminaries results are recalled, such as the generalization of the immediate consequences operator based on generalized quantifiers. Section 3 introduces the operational semantics of the new flexible framework of this paper, and the necessity of the reductants. This last notion is adapted in Section 4, while the completeness results are given in Section 5. Finally, some conclusions and future work are considered in Section 6.

2. Preliminaries

First of all, some definitions and outcomes will be reviewed in this section to help understand the following sections. For more details see [24, 28, 30].

2.1. Generalized quantifiers.

The immediate consequences operator, which will be taken into consideration in this paper, was extended in [30, 31] using generalized quantifiers, in order to weaken the existential feature of the supremum. Therefore, some important issues from generalized quantifiers are recalled next. For more information see [3, 20, 34].

Definition 1. *[34] Let $U = \{u_1, \ldots, u_n\}$ be a finite universe, $\mathcal{P}(U)$ denote the powerset of U and $\mu\colon \mathcal{P}(U) \longrightarrow [0,1]$ be a fuzzy measure, i.e., a monotone mapping with $\mu(\varnothing) = 0$. We say that the fuzzy measure μ is* invariant with respect to cardinality*, if the following condition holds*

$$\text{If } |A| = |B|, \text{ then } \mu(A) = \mu(B), \text{ for all } A, B \in \mathcal{P}(U),$$

where $|\cdot|$ denotes the cardinality of a set.

Now, the definition of generalized quantifier is recalled.

Definition 2. *[34] Let U be a non-empty finite universe, $\mu\colon \mathcal{P}(U) \to [0,1]$ a fuzzy measure invariant with respect to the cardinality and $\otimes$ an operator which is increasing on the first argument and satisfies the following boundary conditions:*[1] *$x \otimes 1 = x$, $x \otimes 0 = 0 \otimes x = 0$, for all $x \in [0,1]$. A mapping $Q_\mu\colon \mathcal{F}(U) \to [0,1]$ defined, for all $C \in \mathcal{F}(U) = \{C \,|\, C\colon U \to [0,1]\}$, as:*

$$Q_\mu(C) = \bigvee_{D \in \mathcal{P}(U)\setminus\{\varnothing\}} \left(\left(\bigwedge_{u \in D} C(u) \right) \otimes \mu(D) \right)$$

is called generalized quantifier determined by the fuzzy measure μ*, where $\bigvee$ and $\bigwedge$ represent the supremum and infimum operators in the unit interval.*

[1] These conditions are required to obtain the universal and existential quantifiers as particular cases of Q_μ.

Next, two of the most significant examples of generalized quantifiers (universal and existential) are shown in the example below, with the corresponding fuzzy measures necessary in order to obtain them.

Example 1. *The universal and existential quantifiers can be obtained from Definition 2 considering the following fuzzy measures $\mu^{\forall}$ and $\mu^{\exists}$, respectively, which are defined as follows:*

$$\mu^{\forall}(D) = \begin{cases} 1 & \text{if} \quad D = U, \\ 0 & \text{if} \quad D \neq U, \end{cases} \qquad \mu^{\exists}(D) = \begin{cases} 0 & \text{if} \quad D = \varnothing, \\ 1 & \text{if} \quad D \neq \varnothing. \end{cases}$$

Furthermore, it can be proved that the following equations are satisfied [6]:

$$Q^{\forall}(C) = \bigwedge_{u \in U} C(u) \qquad Q^{\exists}(C) = \bigvee_{u \in U} C(u).$$

It is important to note that $\mu^{\exists}$ is the maximum fuzzy measure, whereas $\mu^{\forall}$ is the minimum measure of the normalized ones[2]. Eventually, as the generalized quantifier provided by Definition 2 is difficult to compute, the following characterization of generalized quantifiers is recalled next.

Theorem 1. *[34] Let Q_μ be a generalized quantifier in the universe $U = \{u_1, \ldots, u_n\}$ determined by a fuzzy measure μ invariant with respect to the cardinality. Then, for every $C \in \mathcal{F}(U)$,*

$$Q_\mu(C) = \bigvee_{i=1}^{n} C\left(u_{\pi(i)}\right) \otimes \mu\left(\{u_1, \ldots, u_i\}\right),$$

where π is a permutation on $\{1, 2, \ldots, n-1, n\}$ such that $C\left(u_{\pi(n)}\right) \leq C\left(u_{\pi(n-1)}\right) \leq \ldots \leq C\left(u_{\pi(2)}\right) \leq C\left(u_{\pi(1)}\right)$.

2.2. **Multi-adjoint structure and syntax of multi-adjoint logic programs.** The adjoint pair is the basic operator taken into account in multi-adjoint logic programming [28, 29].

Definition 3. *Let $\langle P, \preceq \rangle$ be a poset. An* adjoint pair in $\langle P, \preceq \rangle$ *is a pair of binary operators $(\leftarrow, \&)$ defined on P satisfying that*

1. *The operator $\&$ is increasing in both arguments.*
2. *The operator $\leftarrow$ is increasing in the first argument (consequent) and decreasing in the second one (antecedent).*
3. *The adjoint property holds, that is, for every $x, y, z \in P$ the following equivalence holds:*
$$x \preceq (y \leftarrow z) \text{ if and only if } (x \& z) \preceq y.$$

The following definition sets the algebraic structure containing different adjoint pairs.

Definition 4. *Let $\langle L, \preceq \rangle$ be a lattice. A* multi-adjoint lattice $\mathcal{L}$ *is a tuple $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ satisfying the following conditions:*

1. *$\langle L, \preceq \rangle$ is bounded, i.e., it has a bottom $\bot$ and a top $\top$ element.*
2. *$(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq \rangle$, for all $i \in \{1, \ldots, n\}$.*
3. *$\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$, for all $\vartheta \in L$ and $i \in \{1, \ldots, n\}$.*

Now, we recall the algebraic structure that will be used in this paper, allowing us to take into account additional operators together with those of the multi-adjoint lattice. For more information about the definitions of graded set, Ω-algebra and Ω-subalgebra see [24, 28].

[2]We say that a fuzzy measure μ is normalized if $\mu(U) = 1$.

Definition 5. *Let Ω be a graded set containing operators $\leftarrow_i$ and $\&_i$ for $i \in \{1,\dots,n\}$ and possibly some extra operators, and let $\mathfrak{L} = \langle L, I\rangle$ be an Ω-algebra whose carrier set L is a complete lattice under $\preceq$. We say that $\mathfrak{L}$ is a* multi-adjoint Ω-algebra *with respect to the pairs $(\leftarrow_i, \&_i)$ for $i \in \{1,\dots,n\}$ if $\mathcal{L} = (L, \preceq, I(\leftarrow_1), I(\&_1), \dots, I(\leftarrow_n), I(\&_n))$ is a multi-adjoint lattice.*

Next, we introduce an example of a multi-adjoint Ω-algebra. From now on, the notation $\dot{\nu} = I(\nu)$ will be considered, where $\nu \in \Omega$.

Example 2. *$\langle [0,1], I\rangle$ is a multi-adjoint Ω-algebra with respect to the graded set $\Omega = \{\leftarrow_P, \&_P, \leftarrow_G, \&_G, \wedge_L, @_{2,1}\}$, where $\dot{\&}_P$ and $\dot{\&}_G$ are the product and Gdel t-norms respectively, together with their residuated implications, $\dot{\wedge}_L$ is the Łukasiewicz t-norm and $\dot{@}_{(1,2)}$ is an aggregation operator. These operators are defined for all x, y, $z \in [0,1]$ as follows*

$$\dot{\&}_G(x,y) = \min\{x,y\} \qquad z \dot{\leftarrow}_G y = \begin{cases} 1 & \text{if} \quad y \le z \\ z & \text{if} \quad \text{otherwise} \end{cases}$$

$$\dot{\&}_P(x,y) = x \cdot y \qquad z \dot{\leftarrow}_P y = \begin{cases} 1 & \text{if} \quad y \le z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$

$$\dot{\wedge}_L(x,y) = \max\{0, x+y-1\} \qquad \dot{@}_{(1,2)}(x,y) = \frac{x+2y}{3}.$$

2.3. **Syntax and semantics of multi-adjoint logic programs.** The fundamental notion that allows us to formulate the syntax of multi-adjoint logic programming is that of a *multi-adjoint logic program*, being a set of rules and facts that enables different intelligent systems to be modeled. In order to define such program, a multi-adjoint Ω-algebra $\mathfrak{L}$ will be considered. The definitions of alphabet $A_{\Omega,\Pi}$, Ω-algebra of expressions, well-formed formulas $\mathfrak{F}$ and the homomorphic extension theorem can be found in [28].

Definition 6. *Let Π be a set of propositional symbols. A* multi-adjoint logic program *is a set $\mathbb{P}$ of weighted rules of the form $\langle (A \leftarrow_i \mathcal{B}), \vartheta\rangle$ such that:*

1. *The rule $(A \leftarrow_i \mathcal{B})$ is a formula of $\mathfrak{F}$.*
2. *The confidence factor ϑ is an element (a truth-value) of L.*
3. *The head of the rule A is a propositional symbol of Π.*
4. *The body formula $\mathcal{B}$ is a formula of $\mathfrak{F}$ built from propositional symbols $B_1,\dots,B_n$ ($n \ge 0$) by the use of conjunctors $\&_1,\dots,\&_n$ and $\wedge_1,\dots,\wedge_k$, disjunctors $\vee_1,\dots,\vee_l$ and aggregators $@_1,\dots,@_m$.*

A weighted rule where the body $\mathcal{B}$ is $\top$ is called a *fact*. It must be taken into account that in the Ω-graded set, the element $\top$ is regarded as a constant. Furthermore, the rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ can also be represented as $\langle A \leftarrow @[B_1,\dots,B_n], \vartheta\rangle$, where $@$ is the aggregator defined as a composition of all operators symbols in $\mathcal{B}$, with $B_1,\dots,B_n$ being the propositional symbols appearing in the body $\mathcal{B}$, verifying that $\mathcal{B} = @[B_1,\dots,B_n]$. Now, an example of a multi-adjoint logic program is shown.

Example 3. *In Example 2, identifying the operator symbols in Ω with their interpretations, we can consider the multi-adjoint Ω-algebra*

$$\left\langle [0,1], \le, \leftarrow_G, \&_G, \leftarrow_P, \&_P, \wedge_L, @_{(1,2)}\right\rangle,$$

where $\&_G$ and $\&_P$ are the Gdel and the product conjunctions respectively, with $\leftarrow_G$ and $\leftarrow_P$ being their respective adjoint implications. Moreover, we use the extra operator $\wedge_L$, the Łukasiewicz conjunction and the aggregator $@_{(1,2)}$ given in Example 2. We define the set of propositional symbols Π as:

$$\Pi = \{\texttt{covid}, \texttt{diarrhoea}, \texttt{headache}, \texttt{moderate_fever}, \texttt{cough}, \texttt{waist_pain}\}.$$

We can consider, in an specific situation, the multi-adjoint logic program $\mathbb{P}$ defined through the rules

$$\begin{aligned} \texttt{covid} &\overset{0.8}{\leftarrow_G} @_{(1,2)}\left(\texttt{diarrhoea} \wedge_L \texttt{headache}, \texttt{waist_pain}\right)\\ \texttt{covid} &\overset{0.9}{\leftarrow_P} \texttt{moderate_fever}\\ \texttt{covid} &\overset{0.7}{\leftarrow_P} \texttt{cough}. \end{aligned}$$

These rules are completed with observations about each particular person, in the form of facts. For example, a particular individual may have the symptoms represented by the following facts.

$$\begin{aligned} \texttt{moderate_fever} &\overset{0.6}{\leftarrow} 1\\ \texttt{cough} &\overset{0.8}{\leftarrow} 1\\ \texttt{diarrhoea} &\overset{0.6}{\leftarrow} 1 \end{aligned} \qquad \begin{aligned} \texttt{headache} &\overset{0.7}{\leftarrow} 1\\ \texttt{waist_pain} &\overset{0.3}{\leftarrow} 1. \end{aligned}$$

The concept of interpretation is fundamental in order to provide a meaning to syntactic formulae.

Definition 7. *An* interpretation *is a mapping $I\colon \Pi \to L$ which asigns values of the complete lattice L to the set of propositional symbols. The set of all interpretations of the formulas defined by the Ω-algebra $\mathfrak{F}$ in the Ω-algebra $\mathfrak{L}$ is denoted $\mathcal{I}_{\mathfrak{L}}$.*

Notice that each interpretation $I\colon \Pi \longrightarrow L$ can be extended to the set of formulas F_Ω, i.e. $\hat{I}\colon F_\Omega \longrightarrow L$, using the unique homomorphic extension theorem that was given in [28]. Furthermore, it is possible to extend the ordering $\preceq$ of $\langle L, \preceq\rangle$ to a point-wise ordering in $\mathcal{I}_{\mathfrak{L}}$. This ordering, together with $\mathcal{I}_{\mathfrak{L}}$, forms a complete lattice, denoted by $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq\rangle$ as in [28]. The least interpretation $\triangle$ maps every propositional symbol to the least element $\bot$ of L.

Next, the definition of model and satisfaction is recalled.

Definition 8. *Given an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ is* satisfied *by I if and only if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a* model *of a multi-adjoint logic program $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.*

The following definition recalls the generalization of the immediate consequences operator presented in [12] to the multi-adjoint logic programming framework [28].

Definition 9. *Let $\mathbb{P}$ be a multi-adjoint logic program. The immediate consequences operator $T^{\mathfrak{L}}_{\mathbb{P}}\colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$, mapping interpretations to interpretations, is defined by considering*

$$T^{\mathfrak{L}}_{\mathbb{P}}(I)(A) = \sup\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \,|\, A \overset{\vartheta}{\leftarrow_i} \mathcal{B} \in \mathbb{P}\right\}.$$

From now on, in order to simplify, we will refer to the immediate consequences operator in the multi-adjoint framework just as $T_{\mathbb{P}}$. The next theorem illustrates how models of a logic program are characterized as post-fixed points of the $T_{\mathbb{P}}$.

Theorem 2. *[28] An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a model of a multi-adjoint logic program $\mathbb{P}$ if and only if I is a post-fixed point of $T^M_{\mathbb{P}}$, that is, $T_{\mathbb{P}}(I) \sqsubseteq I$.*

Moreover, this operator is increasing, as the following result shows.

Theorem 3. *[28] The operator $T_{\mathbb{P}}$ is monotonic.*

Since $T_{\mathbb{P}}$ is monotonic, it has a least fixed point [24]. The following result shows that the least fixed point of $T_{\mathbb{P}}$ is equal to the least model of the program, which is denoted as $M_{\mathbb{P}}$. Furthermore, it is possible to compute it by iterating $T_{\mathbb{P}}$ an ordinal number of times, starting from the least interpretation $\triangle$.

Theorem 4. *[28] Let L be a complete lattice and $\mathbb{P}$ a multi-adjoint logic program. We have that $M_{\mathbb{P}} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \,|\, I \text{ is a model of } \mathbb{P}\}$ is a model and, for some ordinal α, it is satisfied that $M_{\mathbb{P}} = lfp\,(T_{\mathbb{P}}) = T^{\alpha}_{\mathbb{P}}(\triangle)$.*

2.4. **Definition of $T_{\mathbb{P}}$ through an arbitrary generalized quantifier.** First of all, we need the following technical definition, in order to present the adaptation of the immediate consequences operator using generalized quantifiers. For more details about this section, see [30, 31]. From now on, a non-empty multi-adjoint logic program $\mathbb{P}$, a set of propositional symbols Π, a multi-adjoint Ω-algebra $\mathfrak{L}$ and a family of fuzzy measures invariant with respect to the cardinality $\mathrm{M} = \{\mu_A \colon \mathcal{P}(\mathbb{P}_A) \to [0,1] \,|\, A \in \Pi, \mathbb{P}_A \neq \varnothing\}$ will be fixed.

Definition 10. *Let I be an interpretation of $\mathbb{P}$ and A a propositional symbol. We define the* mapping $C_I^A \colon \mathbb{P} \longrightarrow [0,1]$*, for each $\langle D \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}$, as follows*

$$C_I^A\left(\langle D \leftarrow_i \mathcal{B}, \vartheta\rangle\right) = \begin{cases} \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) & \text{if } D = A \\ 0 & \text{otherwise .} \end{cases}$$

In addition, this new operator is monotonic with respect to the interpretation I [30].

Now, the general definition of quantified immediate consequences operator is recalled, in which we consider a family of fuzzy measures, where the most beneficial measure can be considered for each propositional symbol.

Definition 11. *[30] Let $\mathbb{P}_A$ be the set of rules of $\mathbb{P}$ with head A. We define the M-quantified immediate consequences operator obtained from the family M as the mapping $T_{\mathbb{P}}^M \colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$ given by*

$$T_{\mathbb{P}}^M(I)(A) = \begin{cases} Q_{\mu_A}\left(C_I^A\right) & \text{if} \quad \mathbb{P}_A \neq \varnothing \\ 0 & \text{if} \quad \mathbb{P}_A = \varnothing \end{cases} \tag{1}$$

for all $A \in \Pi$ and $I \in \mathcal{I}_{\mathfrak{L}}$.

Notice that, if $|\mathbb{P}_A| = 1$, then $\mu = \mu^{\exists}$ and $T_{\mathbb{P}}^M(I)(A) = Q^{\mu_A}(C_I^A) = Q^{\exists}(C_I^A)$. Several illustrative examples can be found in [30]. This new definition is monotonic as the following result recalls.

Proposition 1. *[30] The immediate consequences operator $T_{\mathbb{P}}^M$ is increasing in $\mathcal{I}_{\mathfrak{L}}$.*

In [30], we present some examples on generalized quantifiers, for which a declarative definition of model was given. These definitions were based on solving an specific problem, which can be represented with a quantifier. For example, the declarative definition of μ^1-model was given as "an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ such that, for each $A \in \Pi$, all the rules with head A are satisfied by I, except possibly one of them", which clearly can be understood as the quantifier Q_{μ^1}, and where the fuzzy measure $\mu^1 \colon \mathcal{P}(\mathbb{P}) \to [0,1]$ is defined as:

$$\mu^1(D) = \begin{cases} 0, & \text{if } |D| \leq 1 \\ 1, & \text{otherwise .} \end{cases} \tag{2}$$

However, it is not possible to give a general declarative definition, but only particular for each family of quantifiers. Therefore, aesthetically, the notion of model had to be adapted to the new flexible framework based on the considered generalized quantifier.

Definition 12. *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is an M-model of $\mathbb{P}$, if $Q_{\mu_A}\left(C_I^A\right) \leq I(A)$ for all $A \in \Pi$. In this case, we say that I satisfies the set of rules in $\mathbb{P}$ concerning the family of generalized quantifiers M.*

Clearly, an interpretation M-model of $\mathbb{P}$ it is a post-fixed point of $T_{\mathbb{P}}^{\mathrm{M}}$. The following result is an adaptation of Theorem 4 to this new framework.

Theorem 5. *[30] We have that $M_{\mathbb{P}} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \,|\, I \text{ is an } M\text{-model of } \mathbb{P}\}$ is an M-model and there is an ordinal α such that $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^M\right) = \left(T_{\mathbb{P}}^M\right)^{\alpha}(\triangle)$.*

3. A procedural semantics for multi-adjoint logic programming

The main goal of this section is to provide an algorithm for query answering extending the one given in [29]. Recall that a query (or goal) is a propositional symbol intended as a question $?A$ prompting the system. As a consequence, it can also be used for computing the least M-model of a given program. For this purpose, we will consider the usual definitions, such as the extended language $\mathfrak{F}^e$, and the admissible rules, which will provide a procedural semantics for giving computed answers to queries of programs. Later, we will adapt the notion of correct answer and analyze how this procedure can be considered in the framework of this paper. Recall that we have fixed a non-empty multi-adjoint logic program $\mathbb{P}$, a family of fuzzy measures M, a set of propositional symbols Π and a multi-adjoint Ω-algebra $\mathfrak{L}$.

Definition 13. *[29] Let $V \subseteq [0,1]$ be the set of truth values of the rules in $\mathbb{P}$. The extended language $\mathfrak{F}^e$ is the corresponding Ω-algebra of formulas freely generated from the disjoint union of Π and V. We will refer to the formulas in the language $\mathfrak{F}^e$ simply as* extended formulas *or* e-formulas*. An operator symbol ν interpreted under $\mathfrak{F}^e$ will be denoted as $\bar{\nu}$.*

The procedural semantics is based on the admissible rules [15, 29].

Definition 14. *[29] The* admissible rules *are defined as follows:*

- R1. *Substitute an atom A in an extended formula by $\left(\vartheta \bar{\&}_i \mathcal{B}\right)$ whenever there exists a rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ in $\mathbb{P}$.*
- R2. *Substitute an atom A in an extended formula by 0.*
- R3. *Substitute an atom A in an extended formula by ϑ whenever there exists a fact $\langle A \leftarrow_i \top, \vartheta\rangle$ in $\mathbb{P}$.*

The formal definition of computed answer is given next.

Definition 15. *[29] Let $?A$ be a goal. An element $\dot{@}\,[r_1, \ldots, r_m]$, with $r_i \in L$, for all $i \in \{1, \ldots, m\}$ is said to be a* computed answer for $?A$ (with respect to $\mathbb{P}$) *if there is a sequence $G_0, \ldots, G_{n+1}$ such that*

1. *$G_0 = A$ and $G_{n+1} = \bar{@}\,[r_1, \ldots, r_m]$ where $r_i \in L$ for all $i \in \{1, \ldots, m\}$.*
2. *Every G_i, for $i \in \{1, \ldots, n\}$, is a formula in $\mathfrak{F}'$.*
3. *Every G_{i+1} is inferred from G_i by one of the admissible rules.*

Therefore, the procedural semantics consists of applying admissible rules successively until obtaining an extended formula without propositional symbols $\bar{@}\,[r_1, \ldots, r_m]$. Hence, the computed answer is the interpretation of this formula as an element of $[0,1]$, i.e., $\dot{@}\,(r_1, \ldots, r_m)$.

Example 4. *Considering the program $\mathbb{P}$ presented in Example 3, we will calculate a computed answer for $?$`covid`. For that, we start with $G_0 =$ `covid`. Applying the admissible rule R1 to $\langle$`covid` $\leftarrow_P$ `moderate_fever`$, 0.9\rangle$, we obtain the e-formula $G_1 = 0.9\bar{\&}_P$`moderate_fever` and with the rule R3 applied to the fact $\langle$`moderate_fever` $\leftarrow_P 1, 0.6\rangle$, $G_2 = 0.9\bar{\&}_P 0.6$ is obtained, which does not contain any propositional symbol. Therefore, we can conclude that $\lambda = 0.9\bar{\&}_P 0.6 = 0.54$ is a computed answer for $?$`covid` w.r.t. $\mathbb{P}$.*

Notice that, we could have also started from $\langle$`covid` $\leftarrow_P$ `cough`$, 0.7\rangle$, obtaining the computed answer $\lambda' = 0.7\bar{\&}_P 0.8 = 0.56$. Therefore, different computed answers can be obtained depending on the rules considered for its calculation. □

Next, we must pay attention to the soundness of the procedure. First of all, we need to adapt the classical definition of correct answer.

Definition 16. *An element $\lambda \in L$ is an* M-correct answer *for the program $\mathbb{P}$ and a query $?A$ if for each M-model $I \colon \Pi \longrightarrow L$ of $\mathbb{P}$, the inequality $\lambda \preceq I(A)$ holds.*

Moreover, we can see from Definition 16 that the least value 0 of the lattice is always an M-correct answer. However, it does not provide any information. In order to solve these drawbacks, an outstanding particular case of M-correct answer will be considered.

Definition 17. *We define the* maximum M-correct answer λ^M_{MC}*, for a program* $\mathbb{P}$ *and a query* $?A$ *as*

$$\lambda^M_{MC} = \sup\{\lambda \in [0,1] \mid \lambda \text{ is an M-correct answer for } \mathbb{P} \text{ and } ?A\}.$$

From this definition, by the properties of the supremum, we can deduce that the maximum M-correct answer is unique for each family M. On the other hand, in the following lemma we prove that, indeed, the supremum of M-correct answers is an M-correct answer. Consequently, this lemma provides the basis for the terminology (maximum) used in the previous definition.

Lemma 1. *Let* A *be a propositional symbol, then* λ^M_{MC} *is an M-correct answer for* $\mathbb{P}$ *and the query* $?A$*.*

Proof. Let $\lambda \in [0,1]$ be an M-correct answer for the program $\mathbb{P}$ and the query $?A$. Then, it is satisfied that $\lambda \leq I(A)$, for any M-model I of $\mathbb{P}$. Therefore, by the supremum property, the following inequality is satisfied.

$$\begin{aligned} \lambda^{\mathrm{M}}_{\mathrm{MC}} &= \sup\{\lambda \in L \mid \lambda \text{ is an M-correct answer for } \mathbb{P} \text{ and } ?A\} \\ &\leq I(A). \end{aligned}$$

Consequently, from Definition 16, it is clear that $\lambda^{\mathrm{M}}_{\mathrm{MC}}$ is an M-correct answer for $\mathbb{P}$ and the query $?A$.

□

In the following theorem, we will show that the immediate consequences operator given in Definition 11 is the maximum M-correct answer.

Theorem 6. *Let* A *be a propositional symbol and* α *an ordinal such that* $\left(T^M_{\mathbb{P}}\right)^{\alpha}(\triangle) = lfp(T^M_{\mathbb{P}})$*. Then,* $\left(T^M_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$ *is the maximum M-correct answer for* $\mathbb{P}$ *and* $?A$*, where* $T^M_{\mathbb{P}}$ *is given by Definition 11.*

Proof. From Theorem 5, there exists an ordinal α such that $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)$ is the least M-model of $\mathbb{P}$. Then, for any M-model I of $\mathbb{P}$ it is satisfied that $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A) \leq I(A)$. From Definition 16, if $\lambda \in [0,1]$ is an M-correct answer for $\mathbb{P}$ and the query $?A$, then for every M-model I the inequality $\lambda \leq I(A)$ holds. In particular, for $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)$ it is verified that $\lambda \leq \left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$. In addition, from the definition of supremum, we reach that $\lambda^{\mathrm{M}}_{\mathrm{MC}} \leq \left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$.

On the other hand, as $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A) \leq I(A)$, for all M-model I of $\mathbb{P}$, then $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$ is an M-correct answer for $\mathbb{P}$ and $?A$. And as $\lambda^{\mathrm{M}}_{\mathrm{MC}}$ is the maximum M-correct answer, then $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A) \leq \lambda^{\mathrm{M}}_{\mathrm{MC}}$. Consequently, $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A) = \lambda^{\mathrm{M}}_{\mathrm{MC}}$.

□

As it is said in [29], when we are working in a practical system, continuity is one of the most important properties that the operator $T^{\mathrm{M}}_{\mathbb{P}}$ needs to satisfy, because it implies the existence of such computational model and that it can be reached in numerable iterations ω.

Furthermore, in the example presented by Morishita [33], the program $\mathbb{P}$, composed of the rules $\langle A \leftarrow B, a\rangle$, $\langle A \leftarrow B, b\rangle$ and the fact $\langle B \leftarrow \top, \top\rangle$, is considered, where $\bot < a < \top$ and $\bot < b < \top$. The computed answers are $\bot$, a or b, but as a and b are incomparable, it can be proved that the maximum correct answer for the query $?A$ is $\top$. Therefore, the most interesting answer cannot be computationally computed. This fact motivated the definition of reductant, which will be the central notion of the following section.

4. A new definition of reductants adapted to this framework

This section focuses on the resolution of the problem highlighted above, with the use of reductants. Recall that we have considered a non-empty multi-adjoint logic program $\mathbb{P}$, a family of fuzzy measures M, a set of propositional symbols Π and a multi-adjoint Ω-algebra $\mathfrak{L}$.

Definition 18. *Assume that all the rules in $\mathbb{P}$ with head*[3] *A are $\langle A \leftarrow_{j_i} \mathcal{B}_i, \vartheta_i\rangle$, for $i \in \{1,\dots,l\}$. An* M-reductant *for A is a rule $\langle A \leftarrow @(\mathcal{B}_1,\dots,\mathcal{B}_l),\vartheta\rangle$ such that for any $b_1,\dots,b_l \in [0,1]$ it is satisfied that*

$$Q_{\mu_A}(C_b^A) \;=\; \vartheta \dot{\&} \dot{@}\,(b_1,\dots,b_l)\,,$$

where $(\leftarrow,\&)$ is an adjoint pair and the mapping $C_b^A\colon \mathbb{P}\to[0,1]$, with $b=(b_1,\dots,b_l)$, is defined as

$$C_b^A\left(\left\langle D \leftarrow_i \mathcal{B},\vartheta'\right\rangle\right) = \begin{cases} \vartheta' \dot{\&}_i b_i & \text{if } D = A \\ 0 & \text{otherwise}\,. \end{cases}$$

The following result shows an interesting properties of C_b^A mappings.

Lemma 2. *The mapping $C^A\colon [0,1]^n \to [0,1]^{\mathbb{P}}$, defined as $C^A(b) = C_b^A$, for all $b\in[0,1]^n$, is increasing.*

Proof. The proof is analogous to that given in Proposition 4 in [30].

□

Now, with the following example, we can see that an M-reductant for A can be a fact despite the fact that not all the rules with head A must be facts.

Example 5. *Consider the multi-adjoint logic program $\mathbb{P}_1$ given by the rules $\langle A \leftarrow_{j_1} \top, 1\rangle$ and $\langle A \leftarrow_{j_2} B, 0.1\rangle$, and the measure $\mu^{\exists}$ associated with the existential quantifier. It can be shown that $\langle A \leftarrow_{j_1} \top, 1\rangle$ is an M-reductant for A. Moreover, the rule $\langle A \leftarrow @(\top,B),1\rangle$, where $\dot{@}(b_1,b_2) = \sup\{1\dot{\&}_{j_1}b_1, 0.1\dot{\&}_{j_2}b_2\}$, for all $b_1,b_2\in[0,1]$, is clearly another M-reductant for A.*

The following example shows that different M-reductants also arise properly changing the aggregator operator.

Example 6. *Let $\dot{\&}_P^*\colon [0,1]_{10}\times[0,1]_4\to[0,1]_4$ be a discrete conjunctor [27], where $[0,1]_{10}$ and $[0,1]_4$ are the granular sets of the unit interval into 10 and 4 pieces, respectively. Moreover, we consider the program $\mathbb{P}_2$ consisting of the rules $\langle A\leftarrow_P \top,1\rangle$ and $\langle A\leftarrow_P B, 0.9\rangle$ and the family of fuzzy measures $M=\{\mu_D = \mu^{\exists} \mid D\in\Pi\}$. Then, it can be proved that the following rules are M-reductants for A.*

$$\begin{aligned}
u_1 &= \langle A \leftarrow @(\top,B),1\rangle\,, & u_5 &= \left\langle A \leftarrow_P^* @'(\top,B),1\right\rangle,\\
u_2 &= \langle A \leftarrow_P^* @(\top,B),0.9\rangle\,, & u_6 &= \left\langle A \leftarrow_P^* @''(\top,B),1\right\rangle,\\
u_3 &= \langle A \leftarrow_P^* @(\top,B),1\rangle\,, & u_7 &= \left\langle A \leftarrow_P^* @'''(\top,B),1\right\rangle,\\
u_4 &= \langle A \leftarrow_P^* @(\top,B),0.8\rangle\,, & u_8 &= \left\langle A \leftarrow_P^* @''''(\top,B),1\right\rangle,
\end{aligned}$$

where

$$\begin{aligned}
\dot{@}(b_1,b_2) &= \sup\{1\dot{\&}_P b_1, 0.9\dot{\&}_P b_2\},\\
\dot{@}'(b_1,b_2) &= \sup\{1\dot{\&}_P b_1, 0.9\dot{\&}_P^* b_2\},\\
\dot{@}''(b_1,b_2) &= \sup\{1\dot{\&}_P b_1, 1\dot{\&}_P^* b_2\},\\
\dot{@}'''(b_1,b_2) &= \sup\{1\dot{\&}_P^* b_1, 1\dot{\&}_P b_2\},\\
\dot{@}''''(b_1,b_2) &= \sup\{0.9\dot{\&}_P^* b_1, 1\dot{\&}_P b_2\},
\end{aligned}$$

for all $b_1,b_2\in[0,1]_4$. Therefore, we can see the large number of M-reductants that exist for each A.

The following result shows that the M-models of a program also satisfy the M-reductants.

Proposition 2. *Any M-reductant $\langle A\leftarrow\mathcal{B},\vartheta\rangle$ of $\mathbb{P}$ is satisfied by any M-model of $\mathbb{P}$.*

[3]Note that some rules can be facts, and so some formulas $\mathcal{B}_i$ can be the $\top$ symbol.

Proof. Let I be an M-model of $\mathbb{P}$ and $\langle A \leftarrow \mathcal{B}, \vartheta\rangle$ be an M-reductant of $\mathbb{P}$, with $\mathcal{B} = @(\mathcal{B}_1, \ldots, \mathcal{B}_l)$, where $\langle A \leftarrow_{j_i} \mathcal{B}_i, \vartheta_i\rangle$ are the rules in $\mathbb{P}$ with head A, for all $i \in \{1, \ldots, l\}$. From Definition 12, it holds that $T_{\mathbb{P}}^{\mathrm{M}}(I) \sqsubseteq I$. Consequently,

$$\begin{aligned}
\vartheta \dot{\&} \hat{I}(\mathcal{B}) &= \vartheta \dot{\&} \hat{I}\left(@(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_l)\right) \\
&= \vartheta \dot{\&} \dot{@}\left(\hat{I}(\mathcal{B}_1), \hat{I}(\mathcal{B}_2), \ldots, \hat{I}(\mathcal{B}_l)\right) \\
&\overset{(1)}{=} Q_{\mu_A}\left(C_I^A\right) \\
&= T_{\mathbb{P}}^{M}(I)(A) \\
&\leq I(A),
\end{aligned}$$

where (1) holds from Definition 18 (M-reductant), considering the particular values given by the interpretation I on $\mathcal{B}_i$, with $i \in \{1, \ldots, l\}$. Notice that, if we consider mappings C_I^A, with $I \in \mathcal{I}_{\mathfrak{L}}$ and $A \in \Pi$, it is satisfied that $T_{\mathbb{P}}^{\mathrm{M}}(I)(A) = \vartheta \dot{\&} \dot{@}\left(b_1, \ldots, b_l\right)$, where $b_i = I(\mathcal{B}_i)$, for all $i \in \{1, \ldots, l\}$. □

Following a similar procedure for construction of reductants given in [29], it is also possible to construct M-reductants for any head $A \in \Pi$ of any rule in a given program $\mathbb{P}$.

Definition 19. *The M-reductant built according to this procedure is called* canonical M-reductant for A.

(1) *If all the rules with head A are $\langle A \leftarrow_{j_i} \mathcal{B}_i, \vartheta_i\rangle$ for all $i \in \{1, \ldots, l\}$, where at least one rule is not a fact, an M-reductant for A is any rule $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_l), \top\rangle$ where $\leftarrow$ is any implication with an adjoint conjunctor and $@$ is defined, for all $b = (b_1, \ldots, b_l) \in [0,1]^l$, as*

$$\dot{@}(b_1, \ldots, b_l) = Q_{\mu_A}(C_b^A).$$

(2) *If all the rules with head A are facts $\langle A \leftarrow_{j_i} \top, \vartheta_i\rangle$, for $i \in \{1, \ldots, l\}$, an M-reductant for A is any rule $\langle A \leftarrow \top, \vartheta\rangle$ with $\leftarrow$ any implication with an adjoint conjunctor and $\vartheta = Q_{\mu_A}(C_b^A)$.*

Notice that the canonical M-reductant for a propositional symbol is unique except for the considered implication symbols, although it does not affect the calculations because of its construction.

With the following two examples, we will see how M-reductants are obtained from following Definition 19.

Example 7. *Consider the facts $\langle A \leftarrow_1 \top, 0.2\rangle$ and $\langle A \leftarrow_2 \top, 0.9\rangle$ and the family of fuzzy measures $M = \{\mu_A = \mu^1\}$, which is a singleton. Now, by Definition 19, in order to obtain a canonical M-reductant for A $\langle A \leftarrow \top, \vartheta\rangle$, we need to compute the value ϑ as follows.*

$$\begin{aligned}
\vartheta &\overset{(1)}{=} Q_{\mu_A}(C_b^A) \\
&= \bigvee_{i=1}^{2} C_b^A\left(u_{\pi(i)}\right) \otimes \mu_A\left(\{u_1, \ldots, u_i\}\right) \\
&= \sup\left\{\left(0.9 \&_2 \hat{I}(\top)\right) \otimes \mu_A(\{u_1\}), \left(0.2 \&_1 \hat{I}(\top)\right) \otimes \mu_A(\{u_1, u_2\})\right\} \\
&\overset{(2)}{=} \sup\{0.9 \otimes 0, 0.2 \otimes 1\} \\
&= 0.2,
\end{aligned}$$

where in (1) $b = (\top, \top)$ and (2) holds taking into account that $\mu_A = \mu^1$. Thus, the canonical M-reductant for A is $\langle A \leftarrow \top, 0.2\rangle$.

The following example determines the M-reductants in Example 4.

Example 8. *Taking into account Example 4, and the family of measures* $M = \{\mu_{\texttt{covid}} = \mu^1, \mu_{\texttt{moderate_fever}} = \mu^{\exists}, \mu_{\texttt{cough}} = \mu^{\exists}\}$*, the M-reductant for* `covid` *is* $\langle \texttt{covid} \leftarrow_G @(\texttt{moderate_fever}, \texttt{cough}), \top \rangle$ *where we have that*

$$\dot{@}(b_1, b_2) = Q_{\mu_{\texttt{covid}}}\left(C_b^{\texttt{covid}}\right),$$

for every $b = (b_1, b_2) \in [0,1]^2$*. Moreover, due to the fact that the measures for* `moderate_fever` *and* `cough` *are the existential ones, the M-reductants are the same facts given in the program for them.*

The following result asserts that canonical M-reductants are really M-reductants.

Proposition 3. *Given* $A \in \Pi$*, the canonical M-reductant for* A *is an M-reductant for* A *in* $\mathbb{P}$*.*

Proof. It straightforwardly follows from the construction of $\langle A \leftarrow \top, \vartheta \rangle$ and $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_l), \top \rangle$ in Definitions 18 and 19.

□

Next, we present a new definition of multi-adjoint logic program which only includes the M-reductants for each propositional symbol $A \in \Pi$.

Definition 20. *We define a* reduced (canonical) program by M-reductants of $\mathbb{P}$ *or (canonical) r-program in short, denoted by* $\mathbb{P}_r$ *(resp.* $\mathbb{P}_r^c$*, when canonical reductants are considered), as a multi-adjoint logic program composed of an M-reductant (resp. the canonical M-reductant) for each propositional symbol* $A \in \Pi$*.*

Notice that, given a program $\mathbb{P}$, the canonical r-program $\mathbb{P}_r^c$ is a particular case of r-program. Therefore, every result true in $\mathbb{P}_r$ will also be true in $\mathbb{P}_r^c$. The consideration of $\mathbb{P}_r$ offers a different point of view of the program $\mathbb{P}$. For example, it provides an alternative for computing the M-models of the program, as the following proposition shows.

Proposition 4. *Let* $\mathbb{P}_r$ *be an r-program of* $\mathbb{P}$*. Then,* I *is an M-model of* $\mathbb{P}$ *if and only if* I *is a model of* $\mathbb{P}_r$*.*

Proof. On the one hand, let $I \in \mathcal{I}_{\mathfrak{L}}$ be an M-model of $\mathbb{P}$. Then, from Proposition 2, it holds that I is a model of $\mathbb{P}_r$.

On the other hand, consider a model $I \in \mathcal{I}_{\mathfrak{L}}$ of $\mathbb{P}_r$. Given a propositional symbol A, if no rule with head A exists in $\mathbb{P}$, then no rule exists in $\mathbb{P}_r$ and the inequality $T_{\mathbb{P}}^{\mathrm{M}}(I)(A) \leq I(A)$ trivially holds. Otherwise, we consider the M-reductant $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_l), \vartheta \rangle$ in $\mathbb{P}_r$, and we obtain that

$$\begin{aligned} T_{\mathbb{P}}^{\mathrm{M}}(I)(A) &= Q_{\mu_A}\left(C_I^A\right) \\ &\overset{(1)}{=} \vartheta \dot{\&} \dot{@}(b_1, \ldots, b_l) \\ &= \vartheta \dot{\&} \dot{@}(I(\mathcal{B}_1), \ldots, I(\mathcal{B}_l)) \\ &= T_{\mathbb{P}_r}(I)(A) \\ &\overset{(2)}{\leq} I(A), \end{aligned}$$

where (1) follows from the definition of M-reductant and (2) is verified due to the fact that I is a model of $\mathbb{P}_r$. Thus, $I \in \mathcal{I}_{\mathfrak{L}}$ is an M-model of $\mathbb{P}$.

□

The following result shows the equivalence between the concept of M-correct answer for $\mathbb{P}$ and correct answer for $\mathbb{P}_r$.

Proposition 5. *Consider an r-program* $\mathbb{P}_r$*, a query* $?A$ *and* $\lambda \in [0,1]$*. Then,* λ *is an M-correct answer for* $\mathbb{P}$ *and* $?A$ *if and only if* λ *is a correct answer for* $\mathbb{P}_r$ *and* $?A$*.*

Proof. Given $\lambda \in [0,1]$, the proof follows from the following chain of equivalencies. λ is an M-correct answer for $\mathbb{P}$ and $?A$ if and only if, for all M-model I of $\mathbb{P}$, $\lambda \leq I(A)$, which is equivalent, by Proposition 4, to $\lambda \leq I(A)$ for all model I of $\mathbb{P}_r$, that is, λ is a correct answer of $\mathbb{P}_r$. □

Hence, in order to obtain a correct answer for a query $?A$, it is not possible to consider the original program $\mathbb{P}$, but a reduced one $\mathbb{P}_r$. This also shows the relevance of the reductants. Moreover, the following example shows that $\mathbb{P}$ cannot be considered as the program from which obtain the M-computed answers for $?A$.

Example 9. *Consider a multi-adjoint logic program $\mathbb{P}$ composed of the rules $\langle \mathtt{A} \leftarrow_P \mathtt{B}, 0.7\rangle$ and $\langle \mathtt{A} \leftarrow_P \mathtt{C}, 0.8\rangle$; and the facts $\langle \mathtt{B} \leftarrow_P \top, 0.9\rangle$, $\langle \mathtt{B} \leftarrow_P \top, 0.7\rangle$, $\langle \mathtt{C} \leftarrow_P \top, 0.9\rangle$ and $\langle \mathtt{C} \leftarrow_P \top, 0.5\rangle$. In addition, we consider the family of fuzzy measures $M = \left\{\mu_D = \mu^1 \mid D \in \Pi\right\}$, where μ^1 is defined in* (2).

On the one hand, the possible computed answers are $\lambda_1 = 0,72$, $\lambda_2 = 0.63$, $\lambda_3 = 0.49$ and $\lambda_4 = 0.4$. On the other hand, $\left(T_{\mathbb{P}}^M\right)^2(\triangle) = lfp\left(T_{\mathbb{P}}^M\right)$ is an M-model of $\mathbb{P}$ and it can be proved that $\left(T_{\mathbb{P}}^M\right)^2(\triangle)(\mathtt{A}) = 0.4$. Therefore, as $0.72 \not\leq 0.4$, we have that 0.72 is a computed answer but not an M-correct answer. Then, not all computed answer for a query $?A$ is an M-correct answer for the same query. Indeed, the "quantification" of the set of computed answers with the fuzzy measure μ^1 does not provide an M-correct answer, since we obtain 0.63, which is not lesser than 0.4.

Consequently, we need to adapt the notion of computed answer to capture the new philosophy provided by the use of generalized quantifiers, in which the reductants are the key.

Definition 21. *Let $\mathbb{P}_r$ be an r-program of $\mathbb{P}$ and a query $?A$. We say that $\lambda \in [0,1]$ is a* $\mathbb{P}_r$-computed answer for $?A$, *if it is a computed answer for $?A$ with respect to $\mathbb{P}_r$, that is, using the admissible rules on the M-reductants in $\mathbb{P}_r$.*

The soundness of the new procedure arises from the following result.

Theorem 7. *Let $\mathbb{P}_r$ be an r-program of $\mathbb{P}$ and a query $?A$. Then, every $\mathbb{P}_r$-computed answer for the query $?A$ is an M-correct answer for $\mathbb{P}$ and $?A$.*

Proof. Let λ be a computed answer for $\mathbb{P}_r$ and the query $?A$. We are going to prove that λ is an M-correct answer by complete induction over the length n of the computation sequence of λ.

For $n = 1$, we have applied the admissible rule R2 or R3.

(1) If we apply R2, then the propositional symbol A is not the head of any rule or fact in $\mathbb{P}_r$. The corresponding sequence of e-formulas is $G_0 = A$, $G_1 = \bot$. Hence, $\lambda = \bot$, which is always an M-correct answer.
(2) If we apply R3, then there is a fact $\langle A \leftarrow \top, \vartheta\rangle \in \mathbb{P}_r$, where $\vartheta = Q_{\mu_A}\left(C_b^A\right)$, and the corresponding sequence of e-formulas is $G_0 = A$, $G_1 = \vartheta$. Therefore, $\lambda = \vartheta$. Now, from the definition of M-model (Definition 12), if $I \in \mathcal{I}_{\mathfrak{L}}$ is an M-model, then $T_{\mathbb{P}}^{\mathrm{M}}(I) \sqsubseteq I$ and, consequently, $\lambda = \vartheta = Q_{\mu_A}\left(C_b^A\right) = T_{\mathbb{P}}^{Q_{\mu_A}}(I)(A) \leq I(A)$, where $b = (b_1, \ldots, b_{|\mathbb{P}_A|}) = (I(\mathcal{B}_1), \ldots, I(\mathcal{B}_{|\mathbb{P}_A|}))$. Therefore, λ is an M-correct answer in $\mathbb{P}_r$.

Suppose that the result is true for a length of the computation sequence of λ less than n, where $1 < n$. In this context, there must necessarily be some rule with head A, since otherwise the length of the sequence would be 1. Therefore, we can construct the M-reductant $\langle A \leftarrow @\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right), \vartheta\rangle$, where $@\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right) = @'\left[B_1, \ldots, B_k\right]$, and we have the sequence $G_0 = A$, $G_1 = \vartheta \bar{\&} \bar{@}'\left[B_1, \ldots, B_k\right]$, …, $G_r = \vartheta \bar{\&} \bar{@}'\left[\lambda_1, \ldots, \lambda_k\right] = \lambda$, where the calculation sequence for each λ_i is less than n and, from the induction hypothesis, $\lambda_i \leq I(B_i)$ for all M-model I and

$i \in \{1, \ldots, k\}$. Consequently, we reach the following chain of inequalities.

$$\begin{aligned}
\lambda &= \vartheta\bar{\&}\dot{@}'[\lambda_1, \ldots, \lambda_k] \\
&\overset{(1)}{\leq} \vartheta\bar{\&}\dot{@}'[I(B_1), \ldots, I(B_k)] \\
&= \vartheta\bar{\&}\hat{I}\left(@'[B_1, \ldots, B_k]\right) \\
&= \vartheta\bar{\&}\hat{I}\left(@(\mathcal{B}_1, \ldots, \mathcal{B}_l)\right) \\
&= \vartheta\bar{\&}\dot{@}\left(\hat{I}(\mathcal{B}_1), \ldots, \hat{I}(\mathcal{B}_l)\right) \\
&\overset{(2)}{=} Q_{\mu_A}\left(C_I^A\right) \\
&= T_{\mathbb{P}_r}^{\mathrm{M}}(I)(A) \\
&\overset{(3)}{\leq} I(A),
\end{aligned}$$

where (1) follows from the monotonicity of $\dot{@}$ and $\bar{\&}$ and the induction hypothesis $\lambda_i \leq I(B_i)$, for all $i \in \{1, \ldots, k\}$, (2) holds from the definition of M-reductant and (3) is verified due to the fact that I is an M-model. Thus, λ is an M-correct answer for the query $?A$ with respect to $\mathbb{P}_r$. □

5. Completeness results

The proof of the completeness theorems follows from some technical results. Recall that a non-empty multi-adjoint logic program $\mathbb{P}$, a family of fuzzy measures M, a set of propositional symbols Π and a multi-adjoint Ω-algebra $\mathfrak{L}$ have been fixed.

Lemma 3. *Let α be an ordinal such that $\left(T_{\mathbb{P}}^{M}\right)^{\alpha}(\triangle) = lfp(T_{\mathbb{P}}^{M})$ and $\lambda \in [0,1]$. We have that $\lambda \leq \left(T_{\mathbb{P}}^{M}\right)^{\alpha}(\triangle)(A)$ if and only if λ is an M-correct answer for $\mathbb{P}$ and $?A$.*

Proof. Consider $\lambda \in [0,1]$ verifying that $\lambda \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A)$. From Theorem 5, $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)$ is the least M-model of $\mathbb{P}$. Consequently, it is clear that $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A) \leq I(A)$ for all M-model I of $\mathbb{P}$ and $A \in \Pi$. Therefore, it holds

$$\lambda \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A) \leq I(A),$$

for all M-model $I \in \mathcal{I}_{\mathfrak{L}}$, concluding that λ is an M-correct answer for $\mathbb{P}$ and $?A$.

Suppose now that λ is an M-correct answer for $\mathbb{P}$ and $?A$. As $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)$ is in particular an M-model of $\mathbb{P}$, we obtain that $\lambda \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A)$. □

The following result shows that each iteration of the immediate consequences operator is a $\mathbb{P}_r$-computed answer for $?A$ (Definition 21).

Proposition 6. *Let $\mathbb{P}_r$ be an r-program of $\mathbb{P}$. Then, for every $n \in \mathbb{N}$ and propositional symbol $A \in \Pi$, $\left(T_{\mathbb{P}}^{M}\right)^{n}(\triangle)(A)$ is a $\mathbb{P}_r$-computed answer for $?A$.*

Proof. We are going to prove this result by induction over the number of iterations n.

For $n = 0$ and $A \in \Pi$, applying R2, then a $\mathbb{P}_r$-computed answer for the query $?A$ is $G_1 = \bot = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{0}(\triangle)(A)$.

Suppose that the result is true for $1 \leq n$ and $Q \in \Pi$, that is, $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n}(\triangle)(Q)$ is a $\mathbb{P}_r$-computed answer for all query $?Q$. Consider $A \in \Pi$, we are going to prove that $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n+1}(\triangle)(A)$ is a $\mathbb{P}_r$-computed answer for $?A$. Clearly, if no rule with head A exists in $\mathbb{P}$, then no rule exists in $\mathbb{P}_r$ and we trivially obtain that result. Otherwise, we consider the M-reductant[4],

[4]Notice that, if the particular case $\mathbb{P}_r^c$ is considered instead of $\mathbb{P}_r$, this M-reductant could represent the fact or the rule of the canonical M-reductant for A (Definition 19), depending on whether A has some fact or not. Therefore, admissible rule R1 or R3 will be applied, depending on the case, and a similar procedure can be considered for proving the result.

$\langle A \leftarrow @(\mathcal{B}_1,\ldots,\mathcal{B}_l),\vartheta\rangle \in \mathbb{P}_r$, with $@(\mathcal{B}_1,\ldots,\mathcal{B}_l) = @'[B_1,\ldots,B_k]$, and where $B_1,\ldots,B_k$ are the propositional symbols of the bodies of the rules with head A.

From the induction hypothesis, $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)(B_i)$, which we will denote as β_i, is a $\mathbb{P}_r$-computed answer for the query $?B_i$, for all $i \in \{1,\ldots,k\}$. Consequently, there is a sequence $H_0 = B_i,\ldots,H_{s_i} = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)(B_i) = \beta_i$, for all $i \in \{1,\ldots,k\}$.

In order to prove that $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n+1}(\triangle)(A)$ is a $\mathbb{P}_r$-computed answer for $?A$, we have to construct a sequence of extended formulas beginning with A and ending with $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n+1}(\triangle)(A)$, applying admissible rules successively. Starting with $G_0 = A$ and applying the admissible rule R1 to the M-reductant $\langle A \leftarrow @(\mathcal{B}_1,\ldots,\mathcal{B}_l),\vartheta\rangle$, we obtain $G_1 = \vartheta\bar{\&}\bar{@}'[B_1,\ldots,B_k]$. Now, each B_i can be substituted by β_i computed previously, for all $i \in \{1,\ldots,k\}$. Therefore, we obtain $G_r = \vartheta\bar{\&}\bar{@}'[\beta_1,\ldots,\beta_k]$, where $r = 1 + s_1 + \ldots + s_k$.

Finally, from the homomorphic extension theorem, the following chain of equalities holds.

$$\begin{aligned}
\vartheta\dot{\&}\dot{@}'[\beta_1,\ldots,\beta_k] &= \vartheta\dot{\&}\dot{@}'[\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)(B_1),\ldots,\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)(B_k)] \\
&= \vartheta\dot{\&}\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n}(\triangle)\left(@'[B_1,\ldots,B_k]\right) \\
&= \vartheta\dot{\&}\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n}(\triangle)\left(@(\mathcal{B}_1,\ldots,\mathcal{B}_l)\right) \\
&= \vartheta\dot{\&}\dot{@}\left(\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n}(\mathcal{B}_1),\ldots,\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n}(\triangle)(\mathcal{B}_l)\right) \\
&\overset{(1)}{=} \bigvee_{i=1}^{l}\left(\vartheta_i\dot{\&}_i\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n}(\triangle)\left(\mathcal{B}_{\pi(i)}\right)\right)\otimes\mu_A\left(\{u_1,\ldots,u_i\}\right) \\
&= \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n+1}(\triangle)(A),
\end{aligned}$$

where (1) follows from the definition of M-reductant (Definition 18). Consequently, we can conclude that $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n+1}(\triangle)(A) = \vartheta\dot{\&}\dot{@}'(\beta_1,\ldots,\beta_k)$ is a $\mathbb{P}_r$-computed answer for $A?$. □

For the following results, we need to recall the concept of continuity. We say that an operator T is *Scott-continuous*, or just *continuous*, if it preserves suprema of directed sets, i.e., given a directed set $X \subseteq L$, it is satisfied that $T(\sup X) = \sup\{T(x) \mid x \in X\}$ [28]. We have now the necessary tools to prove the completeness results.

Theorem 8. *If $T_{\mathbb{P}}^{M}$ is continuous, then for each M-correct answer $\lambda \in [0,1]$ for a program $\mathbb{P}$ and a query $?A$ and for all $\varepsilon < \lambda$, there is an $\mathbb{P}_r$-computed answer δ such that $\varepsilon < \delta$.*

Proof. On the one hand, we are going to prove by induction over n that

$$\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle) \sqsubseteq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n+1}(\triangle). \tag{3}$$

For $n = 0$, this result is clear since $\triangle \sqsubseteq T_{\mathbb{P}}^{\mathrm{M}}(\triangle)$ always holds. Suppose that $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n-1}(\triangle) \sqsubseteq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)$ is true for $1 \leq n$. From Proposition 1, $T_{\mathbb{P}}^{\mathrm{M}}$ is monotonic, then

$$T_{\mathbb{P}}^{\mathrm{M}}\left(\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n-1}(\triangle)\right) \sqsubseteq T_{\mathbb{P}}^{\mathrm{M}}\left(\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)\right)$$

On the other hand, as λ is an M-correct answer, then for all M-model I, it is verified that $\lambda \leq I(A)$. Since $T_{\mathbb{P}}^{\mathrm{M}}$ is continuous, then $\mathrm{lfp}(T_{\mathbb{P}}^{\mathrm{M}}) = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\omega}(\triangle)$ and, by Lemma 3, we have that $\lambda \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\omega}(\triangle)(A)$. Taking into account the continuity of $T_{\mathbb{P}}^{\mathrm{M}}$ and Expression (3), we obtain that

$$\varepsilon < \lambda \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\omega}(\triangle)(A) = \sup\left\{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)(A) \mid n \in \mathbb{N}\right\}.$$

Therefore, since $\left\{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^n(\triangle)(A) \mid n \in \mathbb{N}\right\}$ is an increasing sequence of elements in $[0,1]$ with limit greater or equal than λ, then there exists some $\delta \in X$ (or equivalently $n_0 \in \mathbb{N}$), that is $\delta = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n_0}(\triangle)(A)$, such that

$$\varepsilon < \delta \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\omega}(\triangle)(A)$$

which finishes the proof because of, by Proposition 6, $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n_0}(\triangle)(A)$ is an $\mathbb{P}_r$-computed answer. □

Moreover, the following quasi-completeness theorem holds.

Theorem 9. *If $T_{\mathbb{P}}^{M}$ is continuous, then for each M-correct answer $\lambda \in [0,1]$ for a program $\mathbb{P}$ and a query $?A$, there is a sequence of elements $\lambda_n \in [0,1]$, with $\lambda \leq \sup \lambda_n$, such that, for any $n_0 \in \mathbb{N}$, there exists a $\mathbb{P}_r$-computed answer δ verifying $\lambda_{n_0} \leq \delta$.*

Proof. Consider $\lambda_n = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n}(\triangle)(A)$ and the M-correct answer $\lambda \in [0,1]$. As λ is an M-correct answer and $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\omega}(\triangle)$ is an M-model, then

$$\lambda \leq \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\omega}(\triangle)(A) = \sup\left\{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n}(\triangle)(A) \,|\, n \in \mathbb{N}\right\} = \sup \lambda_n.$$

Now, given $n_0 \in \mathbb{N}$, we consider the value $\delta = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{n_0}(\triangle)(A) = \lambda_{n_0}$, which is a $\mathbb{P}_r$-computed answer for $?A$ by Proposition 6.

□

5.1. **Maximum computed answer.** This section introduces the notion of maximum M-computed answer and studies its computation and properties as well.

Definition 22. *Consider an r-program of $\mathbb{P}$, $\mathbb{P}_r$, and a query $?A$. We say that λ is the* maximum $\mathbb{P}_r$-computed answer*, if it is a $\mathbb{P}_r$-computed answer verifying that $\lambda' \leq \lambda$ for all $\mathbb{P}_r$-computed answer λ' for $?A$.*

Notice that, the maximum $\mathbb{P}_r$-computed answer could not exist, because it is possible to obtain an infinite increasing chain of computed answers without a maximum, as we will show in the following example. However, this is not a relevant problem since the completeness results (Theorems 8 and 9) proved that we can be as close as we require to the maximum $\mathbb{P}_r$-computed answer for a query $?A$. Indeed, Proposition 6 showed the relationship between the iterations of $T_{\mathbb{P}}^{\mathrm{M}}$ and the $\mathbb{P}_r$-computed answers for $?A$, which will be complemented with new results at the end of this section. Thus, the termination results of the immediate consequences operator provided in [9, 11] can also give sufficient conditions to ensure the existence of the maximum $\mathbb{P}_r$-computed answer. This remark complements the developments considered, for instance, in [14, 29].

Example 10. *Considering the multi-adjoint lattice introduced in Example 2, and the multi-adjoint logic program $\mathbb{P}$ consisting of the weighted rules $\langle A \leftarrow @_{(1,2)}(B_1, B_2), 1\rangle$, $\langle C \leftarrow_P A, 0.7\rangle$, $\langle B_1 \leftarrow_P C, 0.9\rangle$, $\langle B_1 \leftarrow \top, 0.2\rangle$ and $\langle B_2 \leftarrow \top, 0.8\rangle$. An r-program $\mathbb{P}_r$ for $\mathbb{P}$ has the rules $\langle A \leftarrow @(B_1, B_2), 1\rangle$, $\langle C \leftarrow @'(A), 1\rangle$, $\langle B_1 \leftarrow @''(C, \top), 1\rangle$, and $\langle B_2 \leftarrow \top, 0.8\rangle$, where $\dot{@}(b_1, b_2) = 1\dot{\&}_P\dot{@}_{(1,2)}(b_1, b_2)$, $\dot{@}'(a) = 0.7\dot{\&}_P a$ and $\dot{@}''(b_1, b_2) = \sup\{0.9\dot{\&}_P b_1, 0.2\dot{\&}_P b_2\}$, for all $a, b_1, b_2 \in [0,1]$.*

The procedure for computing the $\mathbb{P}_r$-computed answer for $?A$ is presented next; where the first column shows the extended formula, while the second one shows the evaluation of the $\mathbb{P}_r$-computed answer, if the admissible rule R2 is used on the remaining propositional, in order to stop the computation. For example, in the second row, B_1 and B_2 have been substituted in the formula by 0. Hence, the obtained value is 0. In the third row, C is substituted by 0 and the evaluation produces $\lambda_2 = 0.53$.

Extended formulae	*R2*
$G_0 = A$	$\lambda_0 = 0$
$G_1 = 1\bar{\&}\bar{@}(B_1, B_2)$	$\lambda_1 = 0$
$G_2 = 1\bar{\&}\bar{@}(B_1, 0.8)$	$\lambda_2 = 0.53$
$G_3 = 1\bar{\&}\bar{@}\left(1\bar{\&}\bar{@}''(C, \top), 0.8\right)$	$\lambda_3 = 0.6$
$G_4 = 1\bar{\&}\bar{@}\left(1\bar{\&}\bar{@}''(1\bar{\&}\bar{@}'(A), \top), 0.8\right)$	$\lambda_4 = 0.6$
$G_5 = 1\bar{\&}\bar{@}\left(1\bar{\&}\bar{@}''(1\bar{\&}\bar{@}'(1\bar{\&}\bar{@}(B_1, B_2)), \top), 0.8\right)$	$\lambda_5 = 0.6$
$G_6 = 1\bar{\&}\bar{@}\left(1\bar{\&}\bar{@}''(1\bar{\&}\bar{@}'(1\bar{\&}\bar{@}(B_1, 0.8)), \top), 0.8\right)$	$\lambda_6 = 0.64$
$G_7 = 1\bar{\&}\bar{@}\left(1\bar{\&}\bar{@}''(1\bar{\&}\bar{@}'(1\bar{\&}\bar{@}((1\bar{\&}\bar{@}(C, \top)), 0.8)), \top), 0.8\right)$	$\lambda_7 = 0.66$

Note that, the computation of the $\mathbb{P}_r$-computed answer has not finished after these seven steps because we have still propositional symbols on it. Consequently, the maximum $\mathbb{P}_r$-computed answer is not reached yet. Moreover, we could have been tempted to define the maximum $\mathbb{P}_r$-computed answer as the value λ_7 obtained in G_7 using the rule R2. Nevertheless, λ_7 would not be the maximum $\mathbb{P}_r$-computed answer for $?P$ since, if we continue the computing process, we will obtain another answer λ_j, with $\lambda_7 < \lambda_j$.

The following theorem provides a mechanism for computing the maximum $\mathbb{P}_r$-computed answer.

Theorem 10. *Let $\mathbb{P}_r$ be an r-program of $\mathbb{P}$, and $?A$ a query. If the maximum $\mathbb{P}_r$-computed answer exists, then it is obtained by applying the rule R2 on atoms B for which no rules or facts with head B exists; otherwise the canonical M-reductant will be used.*

Proof. First of all, notice that at most one canonical M-reductant exists for every propositional symbol. Given an r-program $\mathbb{P}_r$ of $\mathbb{P}$, $\lambda \in [0,1]$, the value obtained following the procedure detailed in the statement of the theorem, and $\lambda' \in [0,1]$ a $\mathbb{P}_r$-computed answer for the query $?A$. We are going to prove $\lambda' \leq \lambda$, by induction over the length n in the calculation of the $\mathbb{P}_r$-computed answer λ'.

If $n = 1$, then it has been applied rule R2 or rule R3.

- R2. In this case, the $\mathbb{P}_r$-computed answer for the query $?A$ is $\lambda' = \bot \leq \lambda$.
- R3. In $\mathbb{P}_r$ there exists the M-reductant $\langle A \leftarrow \top, \vartheta' \rangle$, where $Q_{\mu_A}(C_b^A) = \vartheta' \dot{\&} \top = \vartheta'$ for all $b \in [0,1]^l$, with l being the number of rules in $\mathbb{P}$ with head A. Hence, through the sequence of e-formulas $G'_0 = A$, $G'_1 = \vartheta'$, the $\mathbb{P}_r$-computed answer for $?A$ is $\lambda' = \vartheta'$. Now, for the canonical M-reductant (following the procedure given in the statement of the theorem) there exist two options.
  (1) All the rules with head A in $\mathbb{P}$ are facts $\langle A \leftarrow_{j_i} \top, \vartheta_i \rangle$, with $i \in \{1, \ldots, l\}$. The canonical M-reductant for A is $\langle A \leftarrow \top, \vartheta \rangle$, where $\vartheta = Q_{\mu_A}(C_b^A)$, with $b = (b_1, \ldots, b_l) \in [0,1]^l$. Consequently, in this case, $\lambda' = \vartheta' = Q_{\mu_A}(C_b^A) = \vartheta = \lambda$.
  (2) The rules in $\mathbb{P}$ with head A are $\langle A \leftarrow_{j_i} \mathcal{B}_i, \vartheta_i \rangle$, with $i \in \{1, \ldots, l\}$, where some rule is not a fact.[5] The canonical M-reductant for $?A$ is $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_l), \top \rangle$, where $@(b_1, \ldots, b_l) = Q_{\mu_A}(C_b^A)$, for all $b = (b_1, \ldots, b_l) \in [0,1]^l$. Moreover, we have that λ is obtained from the sequence of e-formulas $G_0 = A$, $G_1 = \top \bar{\&} \bar{@}(\mathcal{B}_1, \ldots, \mathcal{B}_l) = \bar{@}'[B_1, \ldots, B_k]$, ..., $G_r = \bar{@}'[\lambda_1, \ldots, \lambda_k] = \lambda$, where λ_i is obtained following the procedure given in the statement of the theorem, for all $i \in \{1, \ldots, k\}$. By the definition of $@'$, there exist $b'_1, \ldots, b'_l \in [0,1]$, such that $\dot{@}'[\lambda_1, \ldots, \lambda_k] = \dot{@}(b'_1, \ldots, b'_l)$. Finally, the following chain of equalities is reached.

$$\begin{aligned} \lambda' &= \vartheta' \\ &= Q_{\mu_A}(C_{b'}^A) \\ &\overset{(*)}{=} \dot{@}(b'_1, \ldots, b'_l) \\ &= \dot{@}'[\lambda_1, \ldots, \lambda_k] \\ &= \lambda, \end{aligned}$$

  where (*) follows from the definition of M-reductant with $b' = (b'_1, \ldots, b'_l)$.

Now, suppose that the result is true for all length of computations less than n, with $1 < n$. In this case, the first admissible rule applied must necessarily be R1. Let $\langle A \leftarrow @\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right), \vartheta \rangle$ be the M-reductant from $\mathbb{P}_r$ for A, where $@\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right) = @'\left[B_1, \ldots, B_k\right]$, with $B_1, \ldots, B_k$ the propositional symbols of the bodies of the rules with head A. On the one hand, the $\mathbb{P}_r$-computed answer λ' is obtained from the sequence of e-formulas $G'_0 = A$, $G'_1 = \vartheta \bar{\&} \bar{@}\left(\mathcal{B}_1, \ldots, \mathcal{B}_l\right) =$

[5] Notice that, an M-reductant can be a fact although not all rules with head A must be facts, as Example 5 shown.

$\vartheta\bar{\&}\bar{@}'[B_1,\ldots,B_k]$, ..., $G'_n = \vartheta\bar{\&}\bar{@}'[\lambda'_1,\ldots,\lambda'_k] = \lambda'$, where λ'_i is the $\mathbb{P}_r$-computed answer for the query $?B_i$, for all $i \in \{1,\ldots,k\}$.

On the other hand, the canonical M-reductant for $?A$ is $\langle A \leftarrow @''(\mathcal{B}_1,\ldots,\mathcal{B}_l), \top\rangle$ and $G_0 = A$, $G_1 = \top\bar{\&}\bar{@}''(\mathcal{B}_1,\ldots,\mathcal{B}_l) = \top\bar{\&}\bar{@}'''[B_1,\ldots,B_k]$, ..., $G_r = \bar{@}'''[\lambda_1,\ldots,\lambda_k] = \lambda$, where λ_i is the value obtained following the procedure given for the query $?B_i$ in the statement of the theorem, for all $i \in \{1,\ldots,k\}$. By the definition of $@'$ and $@'''$, there exist $b_1,\ldots,b_l,b'_1,\ldots,b'_l \in [0,1]$, such that $\bar{@}'[\lambda'_1,\ldots,\lambda'_l] = \dot{@}'(b'_1,\ldots,b'_l)$ and $\bar{@}'''[\lambda_1,\ldots,\lambda_l] = \dot{@}'''(b_1,\ldots,b_l)$ respectively. In addition, by the induction hypothesis, we have that $\lambda'_i \leq \lambda_i$ for all $i \in \{1,\ldots,k\}$, and the following chain of inequalities is satisfied.

$$\begin{aligned}
\lambda' &= \vartheta\bar{\&}\bar{@}'\left[\lambda'_1,\ldots,\lambda'_l\right] \\
&= \vartheta\bar{\&}\dot{@}'\left(b'_1,\ldots,b'_l\right) \\
&\overset{(1)}{=} Q_{\mu_A}\left(C^A_{b'}\right) \\
&\overset{(2)}{\leq} Q_{\mu_A}\left(C^A_{b}\right) \\
&= \dot{@}'''(b_1,\ldots,b_l) \\
&\overset{(3)}{=} \bar{@}'''[\lambda_1,\ldots,\lambda_l] \\
&= \lambda,
\end{aligned}$$

where (1) and (3) follow from the definition of M-reductant and canonical M-reductant respectively, with $b = (\lambda_1,\ldots,\lambda_l)$, $b' = (\lambda'_1,\ldots,\lambda'_l)$; while (2) is satisfied due to the induction hypothesis, Lemma 2 and the monotonicity of the quantifier.

□

Theorem 10 is really important because it says that the maximum $\mathbb{P}_r$-computed answer for a query $?A$ can be obtained from the reduced program $\mathbb{P}^c_r$ applying admissible rules $R1$ and $R3$ whenever possible. Since $|\mathbb{P}^c_r(A)| \leq 1$, for all $A \in \Pi$, then the sequence of the computation of the maximum $\mathbb{P}^c_r$-computed answer is unique.

Theorem 11. *Let $\mathbb{P}_r$ be an r-program of $\mathbb{P}$, $\left(T^M_{\mathbb{P}}\right)^{\alpha}(\triangle)$ the least fixed point of $T^M_{\mathbb{P}}$ and a query $?A$. If the maximum $\mathbb{P}_r$-computed answer for the query $?A$ with respect to $\mathbb{P}$ exists, then it is $\left(T^M_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$.*

Proof. Given an r-program $\mathbb{P}_r$ of $\mathbb{P}$, we are going to prove this result by induction over the length n of the sequence of the computation of the maximum $\mathbb{P}_r$-computed answer λ for the query $?A$ following the mechanism in Theorem 10.

If $n = 1$, then we have applied admissible rule R2 or R3.

- If the rule applied is R2, following Theorem 10, then there are neither rules nor facts with head A. Consequently, $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A) = \bot = \lambda$, which is the maximum $\mathbb{P}_r$-computed answer.
- If R3 is applied, since the length in the computation of λ is $n = 1$, then the canonical M-reductant in $\mathbb{P}^c_r$ is a fact, that is, $\langle A \leftarrow \top, \vartheta\rangle$. From Definition 19, only facts with head A exist in $\mathbb{P}$, that is

  $$\langle A \leftarrow_{j_1} \top, \vartheta_1\rangle, \ldots, \langle A \leftarrow_{j_l} \top, \vartheta_l\rangle$$

  are all the rules with head A in $\mathbb{P}$. Therefore, the $\mathbb{P}^c_r$-computed answer is obtained from the sequence $G_0 = A$, $G_1 = \vartheta = Q_{\mu_A}\left(C^A_b\right)$, for all $b = (b_1,\ldots,b_l) \in [0,1]^l$. Finally, it only remains to prove that $\lambda = \left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$, which arises from the following chain of

equalities.

$$\begin{aligned}
\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A) &\overset{(1)}{=} T_{\mathbb{P}}^{M}\left(\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)\right)(A)\\
&\overset{(2)}{=} Q_{\mu_A}\left(C^{A}_{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)}\right)\\
&= \bigvee_{i=1}^{l} C^{A}_{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)}\left(u_{\pi(i)}\right)\dot{\otimes}\mu_A\left(\{u_1,\ldots,u_i\}\right)\\
&= \bigvee_{i=1}^{l}\left(\vartheta_{\pi(i)}\dot{\&}_{\pi(i)}\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)(\top)\right)\dot{\otimes}\mu_A\left(\{u_1,\ldots,u_i\}\right)\\
&= \bigvee_{i=1}^{l}\left(\vartheta_{\pi(i)}\dot{\&}_{\pi(i)}\top\right)\dot{\otimes}\mu_A\left(\{u_1,\ldots,u_i\}\right)\\
&\overset{(3)}{=} Q_{\mu_A}\left(C_b^A\right)\\
&\overset{(4)}{=} \vartheta\\
&= \lambda,
\end{aligned}$$

where (1) follows from the fact that α is an ordinal such that $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle) = \mathrm{lfp}(T_{\mathbb{P}}^{\mathrm{M}})$, (2) from the definition of the $T_{\mathbb{P}}^{\mathrm{M}}$, (3) holds because of the definition of the generalized quantifier Q_{μ_A}, with $b = \left(\top,\overset{(l)}{\ldots},\top\right)$, and (4) follows from the definition of M-reductant.

Suppose that the result is true for all length of computation less than n, with $1 < n$. Let $\langle A \leftarrow @\left(\mathcal{B}_1,\ldots,\mathcal{B}_l\right),\top\rangle$ be the canonical M-reductant for A, where $@\left(\mathcal{B}_1,\ldots,\mathcal{B}_l\right) = @'\left[B_1,\ldots,B_k\right]$, where B_i, with $i\in\{1,\ldots,k\}$, are the propositional symbols that appear in $@\left(\mathcal{B}_1,\ldots,\mathcal{B}_l\right)$. From the induction hypothesis, it is verified that $\beta_i = \left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(B_i)$ is the maximum $\mathbb{P}_r$-computed answer for the query $?B_i$, for all $i\in\{1,\ldots,k\}$. Consequently, we obtain the chain of extended formulas $G_0 = A, G_1 = \top\bar{\&}@'\left[B_1,\ldots,B_k\right],\ldots,G_r = \top\bar{\&}@'\left[\beta_1,\ldots,\beta_k\right] = \lambda$. Therefore, by Theorem 10, we obtain that λ is the maximum $\mathbb{P}_r$-computed answer for $?A$. Moreover,

$$\begin{aligned}
\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A) &= T_{\mathbb{P}}^{M}\left(\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)\right)(A)\\
&= Q_{\mu_A}\left(C^{A}_{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)}\right)\\
&\overset{(*)}{=} \top\dot{\&}@\left(\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)(\mathcal{B}_1),\ldots,\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)(\mathcal{B}_l)\right)\\
&= \top\dot{\&}\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)\left(@\left(\mathcal{B}_1,\ldots,\mathcal{B}_l\right)\right)\\
&= \top\dot{\&}\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)\left(@'\left[B_1,\ldots,B_k\right]\right)\\
&= \top\dot{\&}@'\left(\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(B_1),\ldots,\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(B_k)\right)\\
&= \top\dot{\&}@'\left[\beta_1,\ldots,\beta_k\right]\\
&= \lambda,
\end{aligned}$$

where (*) is verified due to the definition of canonical M-reductant, where the vector b is given by the interpretation of the mapping $\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)$ through its image, that is, considering the vector $b = \left(\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)(\mathcal{B}_1),\ldots,\widehat{\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}}(\triangle)(\mathcal{B}_l)\right)$. We can conclude that the maximum M-computed answer is $\left(T_{\mathbb{P}}^{\mathrm{M}}\right)^{\alpha}(\triangle)(A)$.

□

Finally, the maximum M-correct and $\mathbb{P}_r$-computed answers for a query $?A$ are related.

Corollary 1. *Let $\mathbb{P}_r$ be an r-program of $\mathbb{P}$ and a query $?A$. Then, the maximum M-correct answer for $\mathbb{P}$ and $?A$ is equal to the maximum $\mathbb{P}_r$-computed answer for $\mathbb{P}$ and $?A$, if it exists.*

Proof. On the one hand, from Theorem 11, the maximum $\mathbb{P}_r$-computed answer for $?A$ is $\lambda = \left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$, where α is an ordinal such that $\left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle) = \mathrm{lfp}(T^{\mathrm{M}}_{\mathbb{P}})$. On the other hand, from Theorem 6, the maximum M-correct answer for $?A$ is $\lambda^{\mathrm{M}}_{\mathrm{MC}} = \left(T^{\mathrm{M}}_{\mathbb{P}}\right)^{\alpha}(\triangle)(A)$. □

6. Conclusions and future work

In this paper, we have continued the work started in [30, 31], regarding the new definition of the immediate consequences operator considering generalized quantifiers. We have focused on the introduction of an operational semantics in this new framework. In particular, the definitions of correct answers and computed answers for a query $?A$ and a program $\mathbb{P}$ have been adapted and the relationships between them have been studied. Furthermore, a new definition of reductant has been presented to obtain a semantically equivalent program from the original one and improving the soundness of the introduced operational semantics. The program reduced by the new notion of reductant has paid a fundamental role in the proposed procedural semantics.

In addition, some completeness results have been presented in the flexible framework considered in this paper, in which the requirement of the continuity of $T^{\mathrm{M}}_{\mathbb{P}}$ has been essential in several results. Moreover, we have proved that the maximum M-correct answer and the maximum M-computed answer coincide, and can be calculated using admissible rules.

As future work, the introduced operational semantics will be applied to real datasets and more properties will be studied in the near future. Furthermore, other flexible approaches will be analyzed for complementing the one considered in this paper and others in the literature. For example, we will study the definition of the introduced quantified immediate consequences operator on general lattices and on the first order logic programming framework of the multi-adjoint setting [14].

References

[1] Amine Belhadi, S.F.W., Kamble, S., Queiroz, M.M. Building supply-chain resilience: an artificial intelligence-based technique and decision-making framework, *Int. J. Prod. Res.*, V.60, N.14, 2022, pp.4487-4507.

[2] Aragón, R.G., Cornejo, M.E., Medina, J., Moreno-García, J., Ramírez-Poussa, E. Decision support system for photovoltaic fault detection avoiding meteorological conditions, *Int. J. Inf. Technol. Decis. Mak.*, V.21, N.03, 2022, pp.911-932.

[3] Cao, N., Holčapek, M., Štěpnička, M. Extensions of fuzzy relational compositions based on generalized quantifiers, *Fuzzy Sets Syst.*, V.339, 2018, pp.73-98.

[4] Castillo, O., Hammouch, Z., Atangana, A., Vargas-De-Leon, C., Allali, K. Preface of the special issue: Fuzzy logic and its application to modeling epidemies: Coronavirus and beyond (special issue), *Appl. Comput. Math.*, V.20, N.1, 2021, pp.3-4.

[5] Chovanec, A., Bartk, R. History-driven template generation in inductive logic programming, *Appl. Comput. Math.*, V.11, N.2, 2012, pp.200-213.

[6] Cornejo, M.E., Díaz-Moreno, J. C., Medina, J. Generalized quantifiers in formal concept analysis, *J. Comput. Appl. Math.*, V.404, 2022, pp.113772.

[7] Cornejo, M.E., Medina, J. Impact Zadeh's theory to algebraic structures. multi-adjoint algebras, *J. Pure Appl. Math.*, V.12, 2021, pp.126-141.

[8] Costa, V., Dellunde, P., Falomir, Z. The logical style painting classifier based on Horn clauses and explanations (l-SHE), *Log. J. IGPL*, V.29, N.1, 2019, pp.96-119.

[9] Damásio, C., Medina, J., Ojeda-Aciego, M. Termination of logic programs with imperfect information: applications and query procedure, *J. Appl. Log.*, V.5, 2007, pp.435-458.

[10] Damásio, C.V., Pereira, L.M. Monotonic and residuated logic programs, In: *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, pp.748-759, Lecture Notes in Artificial Intelligence, Springer, Berlin, V.2143, 2001, 818 p.

[11] Daz-Moreno, J.C., Medina, J., Portillo, J.R. Fuzzy logic programs as hypergraphs. termination results, *Fuzzy Sets Syst.*, V.445, 2022, pp.22-42.

[12] Emden, M.V., Kowalski, R. The semantics of predicate logic as a programming language, *J. ACM*, V.23, N.4, 1976, pp.733-742.
[13] Hájek, P. *Metamathematics of Fuzzy Logic, Trends in Logic*, Kluwer Academic Publishers, Dordrecht, Boston and London, 1998, 307 p.
[14] Julián-Iranzo, P., Medina, J., Ojeda-Aciego, M. On reductants in the framework of multi-adjoint logic programming., *Fuzzy Sets Syst.*, V.317, 2017, pp.27-43.
[15] Julin-Iranzo, P., Moreno, G., Penabad, J. On fuzzy unfolding: A multi-adjoint approach, *Fuzzy Sets Syst.*, V.154, N.1, 2005, pp.16-33.
[16] Julin-Iranzo, P., Moreno, G., Riaza, J.A. Some properties of substitutions in the framework of similarity relations, *Fuzzy Sets Syst.*, V.465, 2023, pp.108510.
[17] Julin-Iranzo, P., Senz-Prez, F. Implementing wordnet measures of lexical semantic similarity in a fuzzy logic programming system, *Theory Pract. Log. Program.*, V.21, N.2, 2021, pp.264-282.
[18] Julin-Iranzo, P., Senz-Prez, F. Proximity-based unification: An efficient implementation method, *IEEE Trans. Fuzzy Syst.*, V.29, N.5, 2021, pp.1238-1251.
[19] Kifer, M., Subrahmanian, V.S. Theory of generalized annotated logic programming and its applications, *J. Log. Program.*, V.12, 1992, pp.335-367.
[20] Klir, G., Yuan, B. *Fuzzy sets and fuzzy logic: Theory and Applications*, Prentice Hall, New Jersey, 1995, 593 p.
[21] Krajči, S., Lencses, R., Vojtáš, P. A comparison of fuzzy and annotated logic programming, *Fuzzy Sets Syst.*, V.144, N.1, 2004, pp.173-192.
[22] Lanzinger, M., Sferrazza, S., Gottlob, G. MV-Datalog -: Effective rule-based reasoning with uncertain observations, *Theory Pract. Log. Program.*, V.22, N.5, 2022, pp.678-692.
[23] Le, V.H. Extending fuzzy linguistic logic programming with negation, *Math.*, V.10, N.17, 2022, pp.3105.
[24] Lloyd, J. *Foundations of Logic Programming*, Springer Verlag, Berlin, 1987, 214 p.
[25] López-Garza, E., Domínguez-Cruz, R. F., Martell-Chávez, F., Salgado-Tránsito, I. Fuzzy logic and linear programming-based power grid-enhanced economical dispatch for sustainable and stable grid operation in eastern Mexico, *Energies*, V.15, N.11, 2022, pp.4069.
[26] Medina, J., Ojeda-Aciego, M., Ruiz-Calviño, J. Formal concept analysis via multi-adjoint concept lattices, *Fuzzy Sets Syst.*, V.160, N.2, 2009, pp.130-144.
[27] Medina, J., Ojeda-Aciego, M., Valverde, A., Vojtáš, P. Towards biresiduated multi-adjoint logic programming, *Lect. Notes Artif. Intell.*, V.3040, 2004, pp.608-617.
[28] Medina, J., Ojeda-Aciego, M., Vojtáš, P. Multi-adjoint logic programming with continuous semantics, *Lect. Notes Artif. Intell.*, V.2173, 2001, pp.351-364.
[29] Medina, J., Ojeda-Aciego, M., Vojtáš, P. A procedural semantics for multi-adjoint logic programming, In: *Progress in Artificial Intelligence, EPIA'01*, pp. 290-297, Lecture Notes in Artificial Intelligence, Springer, Berlin, V.2258, 2001, 418 p.
[30] Medina, J., Torné-Zambrano, J.A. Immediate consequences operator on generalized quantifiers, *Fuzzy Sets Syst.*, V.456, 2022, pp.72-91.
[31] Medina, J., Torné-Zambrano, J.A. Fuzzy logic programming with generalized quantifiers, *Stud. Comput. Intell.*, V.1040, 2023, pp.17-23.
[32] Mohapatra, D., Chakraverty, S., Castillo, O. Numerical investigation of fluid dynamic model in uncertain environment (special issue), *Appl. Comput. Math.*, V.22, N.3, 2023, pp.297-316.
[33] Morishita, S. *A unified approach to semantics of multi-valued logic programs*, Technical Report RT 5006, IBM Tokyo, 1990.
[34] Štěpnička, M., Holčapek, M. Fuzzy relational compositions based on generalized quantifiers, In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 224-233, Communications in Computer and Information Science, Springer International Publishing, Berlin, V.442, 2014, 603 p.
[35] Valdez, F., Castillo, O., Cortes-Antonio, P., Melin, P. Applications of intelligent optimization algorithms and fuzzy logic systems in aerospace: A review. *Appl. Comput. Math.*, V.21, N.3, 2022, pp.233-245.
[36] Vojtáš, P. Fuzzy logic programming, *Fuzzy Sets Syst.*, V.124, N.3, 2001, pp.361-370.
[37] Zadeh, L. A. Fuzzy sets, *Inf. Control*, V.8, 1965, pp.338-353.

Capítulo 6

Operador de consecuencias inmediatas a través de operadores de promedio ponderado ordenado

Título:	Immediate consequences operator through ordered weighted average operators
Autores:	Jesús Medina, José Antonio Torné-Zambrano
Revista:	Iranian Journal of Fuzzy Systems
Factor de impacto JCR:	1,9. Posición 34 de 490 (Q1)
Año:	2023
Categoría:	Mathematics
Publicación:	Junio 2023
DOI:	https://doi.org/10.22111/IJFS.2023.7696

Iranian Journal of Fuzzy Systems

Volume 20, Number 5, (2023), pp. 71-88

Original paper

University of Sistan and Baluchestan

Immediate consequences operator through ordered weighted average operators

J. Medina[1] and J. A. Torné-Zambrano[2]

[1,2] *Department of Mathematics, University of Cádiz, Cádiz, Spain*

jesus.medina@uca.es,joseantonio.torne@uca.es

Abstract

The immediate consequences operator has been a widely studied and used operator for defining the semantics of a logic program. For instance, it has been considered in the fuzzy case for handling datasets with imperfect, imprecise or vague information. The natural generalization of this operator to the mentioned fuzzy framework is based on the supremum operator, which preserves the strict nature of the universal quantifier. As a consequence, errors in the data, which are usual in the uncertainty environment of the considered dataset, can cause loss of information. This is the main reason why this paper makes different generalizations of this operator by using weighted aggregation operators and introducing interesting results.

Keywords: Immediate consequences operator, ordered weighted operators.

1 Introduction

Multi-adjoint logic programming is based on the computational logic, one branch of the logic, which is the field of the mathematics that deals with modeling human language and has been considered in many fields [1, 3, 4, 11, 25]. From the efforts of diverse authors [7, 10, 26] in the field of computational logic, at the beginning of the 70s, logic programming was born, which was a novel conception considering that logic had only been used as a declarative language [2, 16]. In logic programming there are two parts well differentiated. On the one hand, the syntax, being the set of symbols and formulas that allow to represent the language in a natural way. On the other hand, semantics, which allows us to give meaning to formulas based on their syntactic structure, establishing an inference system that allows us to decide which deductions and conclusions are valid.

The syntax of a multi-set logic program is made up of logic programs, which consist of a series of equations $C \leftarrow \mathcal{B}_1, \ldots, \mathcal{B}_n$, called *rules*, that models a given system. Based on these rules and experimental values obtained by observation, a process of diagnosis and prediction of the variables of the system can be carried out. Defining a system of inference to obtain which deductions or consequences are correct is one of the most common problems in logic programming. The multi-adjoint approach refers to a formal framework that has been recently introduced and which generalizes different approaches of fuzzy logic programming [6, 14, 30]. This is the main reason why this fuzzy framework has been considered in this paper. The algebraic structure of this framework is given by a complete lattice together with several conjunctors and implications, which form adjoint pairs. On the other hand, semantics of a logic program is defined as the post-fixed points of the immediate consequences operator $T_{\mathbb{P}}$ [13, 18, 19, 21, 27, 28, 29], which is defined as usual through the supremum of a set of values. Consequently, the existence of small alterations or variations in the data (due to an incorrect or deficient observation) can alter the final outcome. This relevant issue was recently considered in [22, 23] in which generalized quantifiers were considered to solve it. Another interesting alternative to reducing the impact of possible noise in the data, in the computation of the immediate consequences operator associated with a logic program, is the use of ordered weighted averaging (OWA) operators. Therefore, the

Corresponding Author: J. Medina
Received: October 2022; Revised: April 2023; Accepted: June 2023.
https://doi.org/10.22111/IJFS.2023.7696

fundamental objective of this paper will be to contribute with a new definition of the $T_{\mathbb{P}}$ based on OWA operators that can relieve the need to use the supremum in its definition.

The use of the OWA operators is justified. These operators allow us to aggregate the information, which consists of assembling all the values in a single one, being the representative of the rest of them. One of the main reasons for the popularity of the OWA operator, which has been target of study and research in the latest years [15, 17, 31, 32], is the great flexibility it provides for modeling a wide range of situations. Furthermore, it has some interesting properties such as monotonicity, and can also have other ones, such as continuity and idempotence.

The document is organized as follows: preliminary concepts of multi-adjoint structures, multi-adjoint logic programming and OWA operators are recalled in Section 2. In Section 3, we carry out a generalization of the $T_{\mathbb{P}}$ through OWA operators, first using a particular weights vector, and then generalizing the result with a generic weights vector. In Section 4 the continuity of this new operator is studied, necessary in order to obtain the least model of the program in a countable number of iterations. To finish, some conclusions and future work are presented in Section 5.

2 Preliminaries

We will begin recalling some relevant definitions that will be necessary in the next sections [19, 20].

2.1 Multi-adjoint algebra structure

The algebraic structure that we are going to consider in this paper is based on the notion of *adjoint pair* [19, 20].

Definition 2.1. [19] *Let $\langle P, \preceq\rangle$ be a poset and $(\leftarrow, \&)$ a pair of binary operators in P such that*

1. *The operator $\&$ is increasing in both arguments.*
2. *The operator $\leftarrow$ is increasing in the first argument (consequent) and decreasing in the second one (antecedent)*
3. *The adjoint property holds, $x \preceq (y \leftarrow z)$ verifies if and only if $(x \& z) \preceq y$, for all $x, y, z \in P$.*

In these circumstances, we say that $(\leftarrow, \&)$ is an adjoint pair *in $\langle P, \preceq\rangle$.*

Hence, the algebraic structure is composed of a lattice in which different adjoint pairs are defined.

Definition 2.2. [20] *Let $\langle L, \preceq\rangle$ be a complete lattice. A multi-adjoint lattice $\mathcal{L}$ is a tuple $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ satisfying*

1. *$\langle L, \preceq\rangle$ is bounded, i.e., it has a bottom $\bot$ and a top $\top$ elements.*
2. *$(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq\rangle$, for all $i \in \{1, \ldots, n\}$.*
3. *$\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$, for all $\vartheta \in L$ and $i \in \{1, \ldots, n\}$.*

2.2 Multi-adjoint logic programming

This section introduces some of the most important definitions in the framework of multi-adjoint logic programming, with the aim of making this article self-contained and facilitate the understanding of the following sections. For a more in-depth study, see [19, 20]. The first ones recall the usual definitions in logic programming.

Definition 2.3. [19] *A* graded set *is a set Ω with a function which assigns to each element $\omega \in \Omega$ a number $n \geq 0$, called the* arity *of ω. The set of operators from Ω with arity n will be denoted as Ω_n.*

Definition 2.4. [19] *Given a graded set Ω, an* Ω-algebra *$\mathfrak{U}$ is a pair $\langle A, I\rangle$ where A is a non-empty set called the* carrier*, and I is a function which assigns maps to the elements of Ω as follows:*

1. *Each element $\omega \in \Omega_n$, $n > 0$, is interpreted as a map $I(\omega)\colon A^n \to A$, denoted by $\omega_{\mathfrak{U}}$.*
2. *Each element $c \in \Omega_0$, called* constant*, is interpreted as an element $I(c)$ in A, denoted by $c_{\mathfrak{U}}$.*

It is also necessary the notion of substructure of an Ω-algebra.

Definition 2.5. [19] *Given an Ω-algebra $\mathfrak{U} = \langle A, I\rangle$, an* Ω-subalgebra *$\mathfrak{B}$ is a pair $\langle B, J\rangle$ such that $B \subseteq A$ and*

1. $J(c) = I(c)$ *for all* $c \in \Omega_0$.
2. *Given* $\omega \in \Omega_n$, $J(\omega)\colon B^n \longrightarrow B$ *is the restriction of* $I(\omega)\colon A^n \longrightarrow A$ *to the domain* $B^n \subseteq A^n$.

The following definition introduces the algebraic structure which enables to use extra operators.

Definition 2.6. [20] *Let* Ω *be a graded set containing operators* $\leftarrow_i$ *and* $\&_i$, *for* $i \in \{1, \ldots, n\}$, *and possibly some extra operators, and let* $\mathfrak{L} = \langle L, I\rangle$ *be an* Ω*-algebra whose carrier set* L *is a complete lattice under* $\preceq$. *We say that* $\mathfrak{L}$ *is a* multi-adjoint Ω-algebra *with respect to the pairs* $(\leftarrow_i, \&_i)$, *for* $i \in \{1, \ldots, n\}$, *if* $(L, \preceq, I(\leftarrow_1), I(\&_1), \ldots, I(\leftarrow_n), I(\&_n))$ *is a multi-adjoint lattice.*

Example 2.7. *Given a graded set* $\Omega = \{\leftarrow_P, \&_P, \leftarrow_G, \&_G, \wedge_L, @\}$ *and an* Ω*-algebra* $\mathfrak{L} = \langle [0,1], I\rangle$, *where*[1] $\dot{\&}_P$, $\dot{\&}_L$ *and* $\dot{\&}_G$ *are the product, Gödel and Łukasiewicz t-norms respectively, together with their residuated implications, since the tuple* $(L, \preceq, \dot{\leftarrow}_P, \dot{\&}_P, \dot{\leftarrow}_G, \dot{\&}_G)$ *is a multi-adjoint lattice, we have that* $\mathfrak{L}$ *is a multi-adjoint* Ω*-algebra. Recall that the definitions of these operators, for all* x, y, $z \in [0,1]$, *are*

$$\&_G(x,y) = \min\{x,y\} \qquad z \leftarrow_G y = \begin{cases} 1 & \text{if} \quad y \leq z \\ z & \text{otherwise} \end{cases}$$

$$\&_L(x,y) = \max\{0, x+y-1\} \qquad z \leftarrow_L y = \min\{1, 1-y+z\}$$

$$\&_P(x,y) = x \cdot y \qquad z \leftarrow_P y = \begin{cases} 1 & \text{if} \quad y \leq z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$

2.3 Syntax of propositional multi-adjoint logic programming

The syntax encompasses the set of symbols and formulae used to formally represent different statements.

Definition 2.8. [19] *Let* Ω *be a graded set and* Π *a countably infinite set. The* alphabet $A_{\Omega,\Pi}$ *associated with* Ω *and* Π *is defined to be the disjoint union* $\Omega \uplus \Pi \uplus S$, *where* S *is the set of auxiliary symbols "(", ")" and ",".*

Definition 2.9. [19] *Given a graded set* Ω *and alphabet* $A_{\Omega,\Pi}$. *The* Ω-algebra $\mathfrak{E} = \langle A^*_{\Omega,\Pi}, I\rangle$ of expressions *is defined as follows:*

1. *The carrier* A^*_Ω *is the set of strings over* $A_{\Omega,\Pi}$.
2. *The interpretation function* I *satisfies the following conditions for strings* $a_1, \ldots, a_n \in A^*_{\Omega,\Pi}$*:*
   - $c_{\mathfrak{E}} = c$, *where* c *is a constant operation, that is,* $c \in \Omega_0$.
   - $\omega_{\mathfrak{E}}(a_1) = \omega a_1$, *where* ω *is a unary operation, that is,* $\omega \in \Omega_1$.
   - $\omega_{\mathfrak{E}}(a_1, a_2) = (a_1 \omega a_2)$, *where* ω *is a binary operation, that is,* $\omega \in \Omega_2$.
   - $\omega_{\mathfrak{E}}(a_1, \ldots, a_n) = \omega(a_1, \ldots, a_n)$, *where* ω *is an* n*-ary operation, that is,* $\omega \in \Omega_n$ *and* $n > 2$.

The idea of this definition is that the set $A_{\Omega,\Pi}$ provides syntactic structure to the set of symbols Ω. However, not all symbol combinations make sense. For this reason, the following definition is introduced.

Definition 2.10. [19] *Let* Ω *be a graded set,* Π *a countable set of propositional symbols and* $\mathfrak{E}$ *the algebra of expressions corresponding to the alphabet* $A_{\Omega,\Pi}$. *The* well-formed formulas *(in short, formulas) generated by* Ω *over* Π *is the least subalgebra* $\mathfrak{F}$ *of the algebra of expressions* $\mathfrak{E}$ *containing* Π. *The set of formulas, that is the carrier of* $\mathfrak{F}$, *will be denoted by* F_Ω.

Finally, the unique homomorphic extension theorem is recalled.

Theorem 2.11. [9] *Let* Ω *be a graded set,* Π *a set of propositional symbols,* $\mathfrak{F}$ *the corresponding* Ω*-algebra of formulas. Let* $\mathfrak{L}$ *be an arbitrary* Ω*-algebra with carrier* L. *Then, for every function* $J\colon \Pi \longrightarrow L$ *there is a unique homomorphism* $\hat{J}\colon F_\Omega \longrightarrow L$ *such that:*

1. *For all* $p \in \Pi$, $\hat{J}(p) = J(p)$.
2. *For each constant* $c \in \Omega_0$, $\hat{J}(c_{\mathfrak{F}}) = c_{\mathfrak{L}}$.

[1] In order to simplify the notation, the interpretation of the operator symbols ($I(\&)$) will be denoted with a dot in the top ($\dot{\&}$).

3. *For every* $\omega \in \Omega_n$ *with* $n > 0$ *and for all* $F_i \in F_\Omega$ *with* $i \in \{1, \ldots, n\}$

$$\hat{J}\left(\omega_{\mathfrak{F}}(F_1, \ldots, F_n)\right) = \omega_{\mathfrak{L}}\left(\hat{J}(F_1), \ldots, \hat{J}(F_n)\right).$$

From now on, a graded set Ω, a multi-adjoint Ω-algebra $\mathfrak{L}$, a set of propositional symbols Π, and the corresponding Ω-algebra of well-formed formulas freely generated from Π by the operators in Ω, with carrier L, denoted as $\mathfrak{F}$ will be fixed.

The syntax of multi-adjoint logic programming is focused on defining logic programs, which are built from a multi-adjoint Ω-algebra and the set of well-formed formulas $\mathfrak{F}$.

Definition 2.12. [19] *A* multi-adjoint logic program *is a set* $\mathbb{P}$ *of weighted rules of the form* $\langle (A \leftarrow_i \mathcal{B}), \vartheta \rangle$*, also denoted* $A \overset{\vartheta}{\leftarrow} \mathcal{B}$*, such that:*

1. *The rule* $(A \leftarrow_i \mathcal{B})$ *is a formula of* $\mathfrak{F}$*.*
2. *The confidence factor* ϑ *is an element (a truth-value) of* L*.*
3. *The head of the rule* A *is a propositional symbol of* Π*.*
4. *The body formula* $\mathcal{B}$ *is a formula of* $\mathfrak{F}$ *built from propositional symbols* $B_1, \ldots, B_n$ *(*$n \geq 0$*) by the use of conjunctors* $\&_1, \ldots, \&_m$ *and* $\wedge_1, \ldots, \wedge_k$*, disjunctors* $\vee_1, \ldots, \vee_L$ *and aggregators* $@_1, \ldots, @_r$*.*
5. *Facts are rules with body* $\top$*.*

Occasionally, a rule $A \leftarrow_i \mathcal{B}$ will be written as $A \leftarrow @[B_1, \ldots, B_n]$, with $@$ being the aggregation of all operational symbols in $\mathcal{B}$, where $B_1, \ldots, B_n$ are propositional symbols present in $\mathcal{B}$. Therefore, $\mathcal{B} = @[B_1, \ldots, B_n]$.

Example 2.13. *Taking into account Example 2.7 and associating each operator symbol in* Ω *with their respective interpretions*[2]*, the following multi-adjoint* Ω*-algebra can be considered.*

$$\langle [0,1], \leq, \leftarrow_G, \&_G, \leftarrow_P, \&_P, \wedge_L \rangle,$$

where $\&_G$ *and* $\&_P$ *are the Gödel and the product conjunctors respectively, with their respective implications. Furthermore, we utilize the Łukasiewicz conjunctor* $\wedge_L$ *as an additional operator. We consider the set of propositional symbols* Π *as*

$$\Pi = \{\texttt{Spray water}, \texttt{Sunlight}, \texttt{Concentration of } CO_2, \texttt{Temperature}, \texttt{Humidity}\}.$$

The following multi-adjoint logic program $\mathbb{P}$ *can be considered under certain circumstances.*

$$\begin{aligned}
\texttt{Spray water} &\overset{0.8}{\leftarrow_G} \texttt{Temperature} \wedge_L \texttt{Sunlight} \\
\texttt{Spray water} &\overset{0.9}{\leftarrow_P} \texttt{Concentration of } CO_2 \\
\texttt{Spray water} &\overset{0.2}{\leftarrow_P} \texttt{Humidity}
\end{aligned}$$

For each individual different data/measures are obtained, these facts complement the set of rules in the program, which are formally represented as follows.

$$\begin{aligned}
\texttt{Temperature} &\overset{0.6}{\leftarrow_P} 1 \\
\texttt{Sunlight} &\overset{0.8}{\leftarrow_P} 1 \\
\texttt{Concentration of } CO_2 &\overset{0.6}{\leftarrow_P} 1 \\
\texttt{Humidity} &\overset{0.7}{\leftarrow_P} 1
\end{aligned}$$

[2]We will avoid to put the dot on every symbol in order to simplify the notation.

2.4 Semantics of propositional multi-adjoint logic programming

Semantics of multi-adjoint logic programming will enable us to provide significance to the statements considered through their syntactic structure, and establishes an inference system to obtain which deductions or consequences are correct.

Definition 2.14. [19] *An* interpretation *is a mapping $I\colon \Pi \longrightarrow L$ which assigns values of the complete lattice L to the set of propositional symbols. The set of all interpretations of the formulas defined by the Ω-algebra $\mathfrak{F}$ in the Ω-algebra $\mathfrak{L}$ is denoted as $\mathcal{I}_{\mathfrak{L}}$.*

From the homomorphic extension theorem (Theorem 2.11), an extended interpretation $\hat{I}\colon F_\Omega \longrightarrow L$ can be defined on the set of formulas F_Ω from each interpretation $I\colon \Pi \longrightarrow L$. Moreover, the ordering $\preceq$ of the lattice $\langle L, \preceq\rangle$ is naturally extended to $\mathcal{I}_{\mathfrak{L}}$. The set of interpretations $\mathcal{I}_{\mathfrak{L}}$ together with the point-wise ordering forms a complete lattice [19], denoted by $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq\rangle$. The least interpretation $\triangle$ maps every propositional symbol to the least element $\bot$ of L.

Definition 2.15. [19] *Given an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ is satisfied by I if and only if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a model of a multi-adjoint logic program $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.*

Next, we present the definition of the immediate consequences operator [8] in the framework of multi-adjoint logic programming.

Definition 2.16. [19] *Let $\mathbb{P}$ be a multi-adjoint logic program. The immediate consequences operator $T_{\mathbb{P}}^{\mathfrak{L}}\colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$, mapping interpretations to interpretations, is defined by considering*

$$T_{\mathbb{P}}^{\mathfrak{L}}(I)(A) = \sup\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\right\}.$$

As the omission of $\mathfrak{L}$ is not misleading, we will use the notation $T_{\mathbb{P}}$ when we refer to the immediate consequences operator. Notice that the supremum of the definition exists because $\langle L, \preceq\rangle$ is a complete lattice. Next, we recall several important results that will allow us to define the semantics of multi-adjoint logic programming.

Theorem 2.17. [19] *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a model of a multi-adjoint logic program $\mathbb{P}$ if and only if $T_{\mathbb{P}}(I) \sqsubseteq I$.*

Theorem 2.18. [19] *The operator $T_{\mathbb{P}}$ is monotonic.*

Taking into account that $T_{\mathbb{P}}$ is monotonic, together with the fact that $\mathcal{I}_{\mathfrak{L}}$ is a complete lattice, we obtain that $T_{\mathbb{P}}$ has a least fixed point.

Proposition 2.19. [28] *The operator $T_{\mathbb{P}}\colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ has a least fixed point $lfp(T_{\mathbb{P}})$ verifying that*

$$lfp(T_{\mathbb{P}}) \quad = \quad \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}(I) = I\} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}(I) \sqsubseteq I\}.$$

Theorem 2.17 allows us to relate the least model $M_{\mathbb{P}}$ with the least fixed point of the $T_{\mathbb{P}}$. Furthermore, the following result shows that it can be reached by iterating an ordinal number of times α the operator $T_{\mathbb{P}}$, starting from the least interpretation $\triangle$.

Theorem 2.20. [19] *Let L be a complete lattice and $\mathbb{P}$ a multi-adjoint logic program. Then, for some ordinal α it is satisfied that $M_{\mathbb{P}} = lfp(T_{\mathbb{P}}) = T_{\mathbb{P}}^{\alpha}(\triangle)$.*

Example 2.21. *From Example 2.13 and Theorem 2.20, we will explain how the mapping $T_{\mathbb{P}}$ is computed and the least model of a program is obtained.*

$$\begin{aligned}
T_{\mathbb{P}}(\triangle)(\texttt{Spray Water}) &= \sup\left\{0.9\dot{\&}_P\widehat{\triangle}(CO_2), 0.2\dot{\&}_P\widehat{\triangle}(\texttt{Humidity}), 0.8\dot{\&}_G\widehat{\triangle}(\texttt{Temperature} \wedge_L \texttt{Sunlight})\right\}\\
&= 0.\\
T_{\mathbb{P}}^2(\triangle)(\texttt{Spray Water}) &= \sup\left\{0.9\dot{\&}_P\widehat{T_{\mathbb{P}}}(\triangle)(CO_2), 0.7\dot{\&}_P\widehat{T_{\mathbb{P}}}(\triangle)(\texttt{Humidity}), 0.5\dot{\&}_G\widehat{T_{\mathbb{P}}}(\triangle)(\texttt{Temperature} \wedge_L \texttt{Sunlight})\right\}\\
&= \sup\{0.54, 0.49, 0.4\}\\
&= 0.54.
\end{aligned}$$

The next table displays the iteration of the mapping of $T_{\mathbb{P}}$. Notice that the least fixed point is reached in the second iteration. As a consequence, by Theorem 2.20, the least model of $\mathbb{P}$ is exactly $T_{\mathbb{P}}^2(\triangle)$.

	$\triangle$	$T_{\mathbb{P}}(\triangle)$	$T_{\mathbb{P}}^2(\triangle)$	$T_{\mathbb{P}}^3(\triangle)$
Concentration of CO_2	*0*	*0.6*	*0.6*	*0.6*
Humidity	*0*	*0.7*	*0.7*	*0.7*
Sunlight	*0*	*0.8*	*0.8*	*0.8*
Temperature	*0*	*0.6*	*0.6*	*0.6*
Spray Water	*0*	*0*	*0.54*	*0.54*

Furthermore, this example shows that any interpretation greater than the least fixed point of $T_{\mathbb{P}}$ must not be a model of the program.

2.5 Continuity

In the multi-adjoint logic programming framework, the continuity is defined through the Scott-continuity, which is based on the notion of directed set. We recall some important definitions and results, which can be found in [18] and [19], for instance.

Definition 2.22. [18] *Let $\langle L, \preceq\rangle$ be a lattice and $X \subseteq L$. We say that X is directed if each finite subset of X has an upper bound in X.*

Definition 2.23. [18] *Let L be a complete lattice and $f\colon L \to L$ a mapping. We say that f is Scott-continuous if it preserves suprema of directed sets, that is, given a directed set X we have that*

$$f(\sup X) = \sup\{f(x) \mid x \in X\}.$$

A multivariable mapping $g\colon L^n \to L$ is said to be Scott-continuous if it is Scott-continuous in each argument separately.

The following result, typically ascribed to Kleene, provides the main reason why the continuity of the immediate consequences operator is really important.

Proposition 2.24. [18] *Let $\langle L, \preceq\rangle$ be a complete lattice and $T\colon L \to L$ a Scott-continuous mapping. Then, lfp(T) can be reached in numerable iterations ω of the operator T from the least element of the lattice.*

Next, we will consider the specific case of the multi-adjoint framework in order to recall a result we will use later in the continuity section (Section 4).

Definition 2.25. [19] *Let $\mathfrak{F}$ be a language interpreted on a multi-adjoint Ω-algebra $\mathfrak{L}$, and ω any operator symbol in the language. We say that ω is continuous if its interpretation under $\mathfrak{L}(\dot{\omega})$ is continuous in L.*

Lemma 2.26. [19] *Let $\mathbb{P}$ be a program interpreted on a multi-adjoint Ω-algebra $\mathfrak{L}$, and let $\mathcal{B}$ be any body formula in $\mathbb{P}$. Assume that all the operators @ in $\mathcal{B}$ are continuous, let X be a directed set of interpretations, and write $S = \sup X$; then*

$$\hat{S}(\mathcal{B}) = \sup\left\{\hat{J}(\mathcal{B}) \mid J \in X\right\}.$$

One important remark is that in this paper, due to the fact that we are considering OWA operators on the unit interval, the immediate consequences operator will be defined on this linear lattice. As a consequence, the Scott-continuity coincides with the usual left-continuity on real numbers. Therefore, it is not necessary either to study the other interesting notion of continuity on lattices: the lower semi-continuity (LSC) [19], since it also coincides with the left-continuity on real numbers. Thus, since these three notions of continuity coincides, we will only analyze the Scott-continuity of the new operator proposed in this paper.

2.6 Ordered weighted averaging aggregation operators

The generalization of the immediate consequences operator that will be studied in this paper is based on the concept of ordered weighted averaging aggregation operator, *OWA* from now on [15, 17, 31, 32].

Definition 2.27. [32] *An n-dimensional* OWA operator *is a mapping @$\colon [0,1]^n \longrightarrow [0,1]$, associated with a weights vector $W = (w_1, \ldots, w_n)$, verifying that $w_1 + \ldots + w_n = 1$ and $0 \leq w_i \leq 1$, for all $i \in \{1, \ldots, n\}$, such that*

$$@(a_1, \ldots, a_n) = \sum_{i=1}^{n} w_i a_{\pi(i)}, \tag{1}$$

where π is a permutation of $\{a_1, \ldots, a_n\}$ satisfying that $a_{\pi(n)} \leq \ldots \leq a_{\pi(1)}$.

In order to understand the great flexibility and versatility of this operator, in the following definition we present different kinds of it, which will be useful when generalizing the immediate consequences operator.

Example 2.28. *Different particular OWA operators are presented.*

1. *Consider the weights vector* $W^* = (1, 0, \ldots, 0)$. *It can easily be proved that* $@^*(a_1, \ldots, a_n) = \max\{a_i \mid 1 \leq i \leq n\}$. *In this case, we say that* $@^*$ *is the* maximum OWA.
2. *Taking into account the weights vector* $W_* = (0, \ldots, 0, 1)$, *we can obtain that* $@_*(a_1, \ldots, a_n) = \min\{a_i \mid 1 \leq i \leq n\}$. *In this case,* $@_*$ *is called the* minimum OWA. *OWA operators where the weights vector* w *verifies that* $w_j = 1$ *for some* j *and* $w_i = 0$ *for all* $i \neq j$ *are known as* step-OWAs.
3. *Given the weights vector* $W_{av} = (1/n, 1/n, \ldots, 1/n)$, *it is verified that* $@_{av}(a_1, \ldots, a_n) = \dfrac{1}{n}\sum_{i=1}^{n} a_i$. *In this case, the operator* $@_{av}$ *receives the name of* average *of* $\{a_1, a_2, \ldots, a_n\}$.
4. *A* window-OWA *is an OWA operator where the weights vector* $W = (w_1, \ldots, w_n)$ *verifies that there exist* k, $m \in \{1, \ldots, n\}$ *such that* $k + m \leq n + 1$, $w_i = 0$ *for all* $i < k$ *and* $k + m \leq i$, *and* $w_i = \frac{1}{m}$ *for all* $k \leq i < k + m$.

Now, we recall the following result, which shows that the set of OWA operators is bounded by the maximum OWA (upper bound) and the minimum OWA (lower bound).

Proposition 2.29. [32] *For all OWA operator* $@$, *it is verified that*

$$@_*(a_1, \ldots, a_n) \leq @(a_1, \ldots, a_n) \leq @^*(a_1, \ldots, a_n),$$

for all $(a_1, \ldots, a_n) \in [0, 1]^n$.

Finally, the following result shows that the OWA operator is monotonic.

Theorem 2.30. [32] *Let* $@_W$ *be an OWA operator associated with the weights vector* W. *Given two values vectors* $(a_1, \ldots, a_n)$ *and* $(b_1, \ldots, b_n)$, *ordered such that, for each* $j \in \{1, \ldots, n\}$, *it is satisfied that* $a_j \leq b_j$, *then*

$$@_W(a_1, \ldots, a_n) \leq @_W(b_1, \ldots, b_n).$$

2.6.1 Orness measure

The orness measure [15, 32] is an interesting notion for classifying different OWA operators depending on the "distance" from the maximum (which is also called "or" operator). This notion will also be considered in this paper.

Definition 2.31. [32] *(Orness). We define an orness measure as the mapping orness*: $[0, 1]^n \longrightarrow [0, 1]$ *defined as follows*

$$orness(W) = \frac{1}{n-1}\sum_{i=1}^{n}((n - i)w_i).$$

Notice that the orness value will depend on the weights vector W considered. In the following example, we will see the orness value of the different types of OWA operator, given in Definition 2.28.

Example 2.32. *The following statements hold.*

1. *For* $W^* = (1, 0, \ldots, 0)$ *it is verified that* $orness(W^*) = 1$.
2. *For* $W_* = (0, \ldots, 0, 1)$ *it is satisfied that* $orness(W_*) = 0$.
3. *For* $W_{av} = (1/n, 1/n, \ldots, 1/n)$ *the orness measure is* $orness(W_{av}) = 1/2$.
4. *For* $W = (1/2, 1/2, 0, \ldots, 0)$ *it is verified that* $orness(W) = \frac{2n-3}{2n-2}$. *Clearly,* $\frac{1}{2} \leq orness(W)$ *if* $2 \leq n$ *and the orness of* W *converges to* 1, *when the value of* n *scales up.*

The following result shows a sufficient condition to obtain OWA operators with high (or low) orness.

Theorem 2.33. [32] *Let* $@$ *be an OWA operator. If the weights verify that* $w_{n-i+1} \leq w_i$, *then* $\frac{1}{2} \leq orness(@)$. *On the contrary, if the weights verify that* $w_i \leq w_{n-i+1}$, *then* $orness(@) \leq \frac{1}{2}$.

3 Immediate consequences operator generalized through OWA operators

The objective of this section is to give a new definition of the immediate consequences operator that allows reducing the impact of the possible presence of noise in the data. For that, we will use OWA operators. First of all, we will explain the difficulty that arises with the definition of the $T_{\mathbb{P}}$ through the supremum.

Assume that in a specific situation we have that the set, from which the supremum is to be calculated, is composed of the values a_1, a_2 to a_n. These values have been obtained through observation. It is possible that a simple error in the calculation or computation of the data could lead to a value that is significantly higher than the rest of the values. Consequently, the value of the $T_{\mathbb{P}}$ would not be representative, since it is disturbed by the presence of noise in the data. This situation also serves to exemplify the great dependence of the $T_{\mathbb{P}}$ from only one observed data. Therefore, in certain cases, small variations in the number of observed data can lead to large variations in the value of the $T_{\mathbb{P}}$.

From now on, a finite and non-empty multi-adjoint logic program $\mathbb{P}$ over a set of propositional symbols Π, with cardinal $n \in \mathbb{N}^*$, and a multi-adjoint Ω-algebra $\langle [0,1], \leq, \&_1, \ldots, \&_m \rangle$ will be fixed.

First of all, given a propositional symbol A and the interpretation I, we present the technical definition of the mapping C_I^A, in order to generalize the $T_{\mathbb{P}}$ using OWA operators, which also was used in other setting in [23].

Definition 3.1. [23] *(*Mapping C_I^A*). Let I be an interpretation on $\mathbb{P}$. We define the mapping $C_I^A \colon \mathbb{P} \longrightarrow [0,1]$, for each $\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}$, as follows*

$$C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle\right) = \begin{cases} \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) & \text{if } C = A \\ 0 & \text{otherwise} \end{cases}$$

These mappings are monotonic on the set of interpretations $\mathcal{I}_{\mathfrak{L}}$ [23, Proposition 4]. In the following subsection, we are going to present the generalization of the immediate consequences operator, given by Definition 2.16, through OWA operators.

3.1 Extending the $T_{\mathbb{P}}$ through an arbitrary OWA operator

Taking into account Definition 2.27, we present the following generalization of the $T_{\mathbb{P}}$ using OWA operators.

Definition 3.2. *(*OWA immediate consequences operator*). Let $W \in [0,1]^n$ be a weights vector, with $\sum_{i=1}^{n} w_i = 1$. The* OWA immediate consequences operator $T_{\mathbb{P}}^W \colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ *is defined, for all $I \in \mathcal{I}_{\mathfrak{L}}$ and $A \in \Pi$, as*

$$T_{\mathbb{P}}^W(I)(A) \quad = \quad @_W\left(C_I^A(u_1), \ldots, C_I^A(u_n)\right), \tag{2}$$

where the mapping C_I^A is given by Definition 3.1 and $\{u_1, \ldots, u_n\}$ is the set of rules of $\mathbb{P}$.

The following result shows that $T_{\mathbb{P}}^W$ is really a generalization of the immediate consequences operator.

Proposition 3.3. *It is satisfied that*

$$T_{\mathbb{P}}^{W^*}(I)(A) = T_{\mathbb{P}}(I)(A),$$

where $W^* = (1, 0, \ldots, 0)$ (Example 2.32) and C_I^A is the mapping given by Definition 3.1.

Proof. Given an interpretation I, a propositional symbol A, and the set of rules with head A, $\mathbb{P}_A$, we obtain that

$$\begin{aligned} T_{\mathbb{P}}^{W^*}(I)(A) &= @_{W^*}\left(C_I^A(u_1), \ldots, C_I^A(u_n)\right) \\ &= \max\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\right\} \\ &\overset{(*)}{=} T_{\mathbb{P}}(I)(A), \end{aligned}$$

where $(*)$ holds because of the unit interval is totally ordered and $\mathbb{P}$ is finite. Notice that, if $\mathbb{P}_A = \varnothing$, then $C_I^A(u) = 0$, for all $u \in \mathbb{P}$ and trivially $T_{\mathbb{P}}^{W^*}(I)(A) = 0 = T_{\mathbb{P}}(I)(A)$ holds. □

Therefore, if the maximum OWA is considered, then the classical definition of the immediate consequences operator in a fuzzy framework is obtained (Definition 2.16). Next, we prove that $T_{\mathbb{P}}^W$ is monotonic.

Proposition 3.4. *The operator $T_{\mathbb{P}}^W$, given by Definition 2, is increasing in $\mathcal{I}_{\mathfrak{L}}$.*

Proof. Given two interpretations I_1, $I_2 \in \mathcal{I}_{\mathfrak{L}}$, such that it is verified that $I_1 \sqsubseteq I_2$, we have to prove that $T_{\mathbb{P}}^{W}(I_1) \sqsubseteq T_{\mathbb{P}}^{W}(I_2)$. Given $A \in \Pi$ and $\mathbb{P} = \{u_1, \ldots, u_n\}$, we have that

$$\begin{aligned} T_{\mathbb{P}}^{W}(I_1)(A) &= @_W\left(C_{I_1}^{A}(u_1), \ldots, C_{I_1}^{A}(u_n)\right) \\ &\overset{(*)}{\leq} @_W\left(C_{I_2}^{A}(u_1), \ldots, C_{I_2}^{A}(u_n)\right) \\ &= T_{\mathbb{P}}^{W}(I_2)(A), \end{aligned}$$

where $(*)$ holds because of the mappings C_I^A are increasing on $\mathcal{I}_{\mathfrak{L}}$, by Theorem 2.30. Thus, $T_{\mathbb{P}}^{W}$ is increasing. □

Next, we will study a certain example of the operator $T_{\mathbb{P}}^{W}$ for a better comprehension of its definition, with the aim of characterizing the weights vectors to generalize the immediate consequences operator for solving the problem of the possible presence of noise in the data. In this case, we are going to consider a weights vector in order to stay with the arithmetic mean of the two greatest values.

Example 3.5. *Assuming that the cardinal of $\mathbb{P}$ is greater or equal than 2, we can consider the following particular OWA immediate consequences operator*

$$T_{\mathbb{P}}^{W_2}(I)(A) = @_{W_2}\left(C_I^A(u_1), \ldots, C_I^A(u_n)\right),$$

where the mapping C_I^A is given by Definition 3.1, and $W_2 = \left(\frac{1}{2}, \frac{1}{2}, 0, \overset{(n-2)}{\ldots}, 0\right)$. The OWA operator with the weights vector given by W_2 is a particular case of window-OWA.

The usual next step is to analyze the semantics of the new immediate consequences operator $T_{\mathbb{P}}^{W}$. Before that, we will studied the specific case of $T_{\mathbb{P}}^{W_2}$, where the weights vector W_2 was introduced in Example 3.5, in order to facilitate the comprehension of the semantics in the general case.

3.2 The fixed point semantics associated with $T_{\mathbb{P}}^{W_2}$

Since the definition of $T_{\mathbb{P}}$ has been relaxed, the notion of model must be adapted to the new philosophy. In this section we will be focused on the particular weight W_2, and we will assume that $2 \leq |\mathbb{P}|$.

Definition 3.6. *(W_2-model). We say that an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a W_2-model of the program $\mathbb{P}$ if all the rules are satisfied by I except possibly one of them, in that case it is verified that the mean of the two highest values $\vartheta \hat{\&} \hat{I}(\mathcal{B})$, with $\langle A \leftarrow \mathcal{B}, \vartheta \rangle \in \mathbb{P}$, is less than the value of the interpretation in the propositional symbol A.*

The following theorem presents the characterization of W_2-models of a program $\mathbb{P}$ as the post-fixed points of $T_{\mathbb{P}}^{W_2}$.

Theorem 3.7. *An interpretation I is a W_2-model for $\mathbb{P}$ if and only if $T_{\mathbb{P}}^{W_2}(I) \sqsubseteq I$.*

Proof. Given the set of rules $\{u_1, \ldots, u_n\}$ of $\mathbb{P}$, and the permutation π considered by the OWA operator (Definition 2.27). If all the rules are satisfied by I, then in particular it is verified that $C_I^A\left(u_{\pi(1)}\right)$, $C_I^A\left(u_{\pi(2)}\right) \leq I(A)$ and, therefore

$$T_{\mathbb{P}}^{W_2}(I)(A) = \frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A).$$

On the contrary, if some rule is not satisfied by I, then it must necessarily be $u_{\pi(1)}$. But, as I is a W_2-model, from Definition 3.6, we obtain that

$$T_{\mathbb{P}}^{W_2}(I)(A) = \frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A).$$

Now, in order to prove the reciprocal, suppose that the interpretation I satisfies that $T_{\mathbb{P}}^{W_2}(I) \sqsubseteq I$. Taking into account that $C_I^A\left(u_{\pi(2)}\right) \leq C_I^A\left(u_{\pi(1)}\right)$, it is verified that

$$C_I^A\left(u_{\pi(n)}\right) \leq \ldots \leq C_I^A\left(u_{\pi(2)}\right) \leq \frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A).$$

Consequently, all the rules of $\mathbb{P}$ are satisfied by I except possibly $u_{\pi(1)}$, in which case it is verified that

$$\frac{C_I^A\left(u_{\pi(1)}\right) + C_I^A\left(u_{\pi(2)}\right)}{2} \leq I(A).$$

□

We can characterize the least W_2-model as the least fixed point of the operator $T_{\mathbb{P}}^{W_2}$, using Theorem 3.7. First of all, the following proposition holds.

Proposition 3.8. *The interpretation* $M_{\mathbb{P}}\colon \Pi \longrightarrow [0,1]$ *defined as* $M_{\mathbb{P}} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } W_2\text{-model of} \mathbb{P}\}$ *is a* W_2*-model of the program* $\mathbb{P}$*, which is the least* W_2*-model of* $\mathbb{P}$ *and* $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{W_2}\right)$.

Proof. In the first place, we can consider the infimum of the set

$$\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } W_2\text{-model of } \mathbb{P}\},$$

which exists because of $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$ is a complete lattice. As $T_{\mathbb{P}}^{W_2}$ is a monotonic mapping, then by Proposition 2.19 and Theorem 3.7 it is verified that

$$\begin{aligned} \operatorname{lfp}\left(T_{\mathbb{P}}^{W_2}\right) &= \inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}^{W_2}(I) \sqsubseteq I\right\} \\ &= \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } W_2\text{-model of } \mathbb{P}\} = M_{\mathbb{P}}. \end{aligned}$$

Therefore, by the definition of fixed point, it is verified that $T_{\mathbb{P}}^{W_2}\left(\operatorname{lfp}\left(T_{\mathbb{P}}^{W_2}\right)\right) = \operatorname{lfp}\left(T_{\mathbb{P}}^{W_2}\right)$, and so $\operatorname{lfp}\left(T_{\mathbb{P}}^{W_2}\right)$ is a W_2-model of $\mathbb{P}$ from Theorem 3.7. In conclusion, $M_{\mathbb{P}}$ is a W_2-model of $\mathbb{P}$, being the least W_2-model of $\mathbb{P}$. □

Lastly, the generalization of Theorem 2.20 is given.

Theorem 3.9. *For some ordinal* α*, it is satisfied that* $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{W_2}\right) = T_{\mathbb{P}}^{\alpha}(\triangle)$.

Proof. This result can easily be proved from Proposition 3.8 and Theorem 2.20. □

3.3 Fixed-point semantics of the general case $T_{\mathbb{P}}^{W}$

In this section, we will introduce the fixed-point semantics of a multi-adjoint logic program considering an arbitrary OWA immediate consequences operator through a general weights vector W.

Owing to the intricateness of its syntax (Definition 3.2), it is not easy to define a notion of model for every vector W. As a consequence, considering the results presented in Section 3.1, the following general definition of model related to a weights vector W is presented.

Definition 3.10. *(*W-model*). We say that an interpretation* $I \in \mathcal{I}_{\mathfrak{L}}$ *is a* W-model *of a multi-adjoint logic program* $\mathbb{P}$ *if and only if it is a post-fixed point of* $T_{\mathbb{P}}^{W}$.

From the previous definition, it is clear that the least W-model of the program $\mathbb{P}$ is the least fixed point of $T_{\mathbb{P}}^{W}$. Furthermore, it is satisfied that the set of W-models with the point-wise order forms a complete lattice.

3.4 Generalization of $T_{\mathbb{P}}$ on a family of OWA operators

The sections above have conveniently generalized the immediate consequences operator through OWA operators, providing a more flexible fixed-point semantics for the obtention of information of data sets. Nevertheless, a problem related to the number of rules with head a propositional symbol can arise.

Assume that the program $\mathbb{P}$ is composed of a single rule $\langle A \leftarrow B, \vartheta \rangle$, i.e., $|\mathbb{P}| = 1$, and let $T_{\mathbb{P}}^{W_1}$ be the OWA immediate consequences operator associated with the vector $W_1 = (0, 1, 0, \ldots, 0)$. As we have only one rule, then the value of $T_{\mathbb{P}}^{W_1}(I)(A)$ would always be 0, for all $I \in \mathcal{I}_{\mathfrak{L}}$. Therefore, we have a problem with those propositional symbols heads of a single rule. One possible solution is to consider a weights vector given by

$$\tau(A) = \begin{cases} 1 & \text{if} \quad |\mathbb{P}_A| = 1 \\ (\alpha_1, \alpha_2) & \text{if} \quad |\mathbb{P}_A| = 2, \quad \text{with} \quad \alpha_1 + \alpha_2 = 1 \\ \left(\alpha_1, \alpha_2, \alpha_3, \overset{(n)}{\ldots}, \alpha_n\right) & \text{if} \quad |\mathbb{P}_A| = n \geq 3, \\ & \quad \text{with} \quad \alpha_1 + \ldots + \alpha_n = 1 \end{cases} \tag{3}$$

where $\mathbb{P}_A$ is the subset of rules with head A. For example, we can consider the following particular OWA immediate consequences operator

$$T_{\mathbb{P}}^{\tau}(I)(A) = @_{\tau(A)}\left(C_I^A(u_1), \ldots, C_I^A(u_n)\right),$$

where $\mathbb{P}_A = \{u_1, \ldots, u_n\}$ and $\tau \colon \Pi \longrightarrow \left\{(1), \left(\frac{1}{2}, \frac{1}{2}\right), \ldots, \left(\frac{1}{2}, \frac{1}{2}, 0, \overset{(n-2)}{\ldots}, 0\right)\right\}$ is defined as

$$\tau(A) = \begin{cases} 1 & \text{if} \quad |\mathbb{P}_A| = 1 \\ \left(\frac{1}{2}, \frac{1}{2}\right) & \text{if} \quad |\mathbb{P}_A| = 2 \\ \left(\frac{1}{2}, \frac{1}{2}, 0, \overset{(m-2)}{\ldots}, 0\right) & \text{if} \quad 3 \leq |\mathbb{P}_A| = m \end{cases} \tag{4}$$

Clearly this new definition is increasing in $\mathcal{I}_{\mathfrak{L}}$. The OWA operator with the weights vector given by τ is a particular case of window-OWA.

Observe that in this case we have a clear distinction between $|\mathbb{P}_A| = 1$ and $|\mathbb{P}_A| > 1$. This idea is also captured with the following definition, which provides with a more versatile solution based on considering a weights vector for each propositional symbol.

Definition 3.11. *($\mathcal{W}$-*OWA immediate consequences operator*). Given a family of weights vectors*[3]

$$\mathcal{W} = \left\{ W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1, A \in \Pi, \mathbb{P}_A \neq \varnothing, |\mathbb{P}_A| = n_A \right\},$$

where $n_A \in \mathbb{N}^*$. We define the $\mathcal{W}$-OWA immediate consequences operator obtained from the family $\mathcal{W}$ *as the mapping $T_{\mathbb{P}}^{\mathcal{W}} \colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$ defined by*

$$T_{\mathbb{P}}^{\mathcal{W}}(I)(A) = \begin{cases} @_{W_A}\left(C_I^A(u_1), \ldots, C_I^A(u_{n_A})\right) & \text{if} \quad \mathbb{P}_A \neq \varnothing \\ 0 & \text{if} \quad \mathbb{P}_A = \varnothing \end{cases}. \tag{5}$$

where $\mathbb{P}_A = \{u_1, \ldots, u_{n_A}\}$, for all $A \in \Pi$ and $I \in \mathcal{I}_{\mathfrak{L}}$.

In the following result, we present the relation between the previous definition and Definition 2.

Proposition 3.12. *Let $\mathbb{P}$ be a multi-adjoint logic program and W a weights vector, then there exists a family of weights vectors*

$$\mathcal{W} \quad = \quad \left\{ W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1,\ A \in \Pi,\ \mathbb{P}_A \neq \varnothing, |\mathbb{P}_A| = n_A \right\},$$

such that $T_{\mathbb{P}}^{W} = T_{\mathbb{P}}^{\mathcal{W}}$.

Proof. Given a weights vector W, we can consider the family of weights vectors

$$\mathcal{W} = \left\{ W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1,\ A \in \Pi,\ \mathbb{P}_A \neq \varnothing, |\mathbb{P}_A| = n_A \right\},$$

where the weights vector W_A is defined as $W_A[i] = W[i]$, for all $1 \leq i \leq n_A$, $A \in \Pi$. Since $\sum_{i=1}^{n} W[i] = 1$, then clearly $\sum_{i=1}^{n_A} W_A[i] \leq 1$, for all $A \in \Pi$.

Given $A \in \Pi$, if $|\mathbb{P}_A| = 0$, then the equality straightforwardly holds, because of $T_{\mathbb{P}}^{\mathcal{W}}(I)(A) = 0$, by definition, and $T_{\mathbb{P}}^{W}(I)(A) = 0$, by the boundary condition of the aggregator operator with the bottom element.

Otherwise, the set $\mathbb{P}_A$ is not empty and, given π_A and π the corresponding permutations, the following equalities holds for every interpretation $I \in \mathcal{I}_{\mathfrak{L}}$.

$$\begin{aligned} T_{\mathbb{P}}^{\mathcal{W}}(I)(A) &= T_{\mathbb{P}}^{W_A}(I)(A) = \sum_{i=1}^{n_A} W_A[i] C_I^A(u_{\pi_A(i)}) = \sum_{i=1}^{n_A} W[i] C_I^A(u_{\pi_A(i)}) + \sum_{i=n_A}^{n} W[i] \cdot 0 \\ &= \sum_{i=1}^{n_A} W[i] C_I^A(u_{\pi(i)}) + \sum_{i=n_A}^{n} W[i] C_I^A(u_{\pi(i)}) = \sum_{i=1}^{n} W[i] C_I^A(u_{\pi(i)}) \\ &= T_{\mathbb{P}}^{W}(I)(A). \end{aligned}$$

[3] $W[i]$ denotes the i-th component of the vector W. Moreover, notice that the definition of OWA can trivially be adapted to consider weights vectors verifying $\sum_{i=1}^{n_A} W_A[i] \leq 1$.

□

This new operator $T_{\mathbb{P}}^{\mathcal{W}}$ is also increasing, as it is proved by the next proposition. From now on, a family of weights vectors $\mathcal{W} = \{W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1, A \in \Pi, \mathbb{P}_A \neq \varnothing, |\mathbb{P}_A| = n_A\}$ will be fixed.

Proposition 3.13. *The immediate consequences operator $T_{\mathbb{P}}^{\mathcal{W}}$ introduced in Definition 3.11 is increasing in $\mathcal{I}_{\mathfrak{L}}$.*

Proof. The proof is similar to the one given in Proposition 3.4. □

Consequently, we have presented a new definition of immediate consequences operator based on a family of weights vectors, which verifies interesting properties such as the monotonicity.

Analogously as in Definition 3.10, the next definition of model related to a family of weights vectors W can be established.

Definition 3.14. *($\mathcal{W}$-model). An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a $\mathcal{W}$-model of a multi-adjoint logic program $\mathbb{P}$ if and only if it is a post-fixed point of $T_{\mathbb{P}}^{\mathcal{W}}$.*

It can be directly shown that the set of $\mathcal{W}$-models, along with the point-wise ordering, forms a complete lattice on $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$.

As in the classical case (Proposition 2.19), by the monotonicity of $T_{\mathbb{P}}^{\mathcal{W}}$ (Proposition 3.13), this operator has a least fixed point $\mathrm{lfp}\big(T_{\mathbb{P}}^{\mathcal{W}}\big)$ verifying that

$$\mathrm{lfp}\left(T_{\mathbb{P}}^{\mathcal{W}}\right) = \inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}^{\mathcal{W}}(I) \sqsubseteq I\right\}.$$

The following result shows the expected characterization of the least $\mathcal{W}$-model as the least fixed point of the operator $T_{\mathbb{P}}^{\mathcal{W}}$.

Proposition 3.15. *The interpretation $M_{\mathbb{P}} \colon \Pi \longrightarrow [0,1]$ given by*

$$M_{\mathbb{P}} = \inf\left\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } \mathcal{W}\text{-model of } \mathbb{P}\right\},$$

is a $\mathcal{W}$-model of $\mathbb{P}$, which is the least $\mathcal{W}$-model of $\mathbb{P}$ and $M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{\mathcal{W}}\right)$.

Proof. This result can be proved analogously to the Proposition 3.8.

□

The next result is the generalization of Theorem 2.20 adapted to the new setting.

Theorem 3.16. *Let $\mathbb{P}$ be a multi-adjoint logic program. Then, for some ordinal α, it is satisfied that*

$$M_{\mathbb{P}} = lfp\left(T_{\mathbb{P}}^{\mathcal{W}}\right) = \left(T_{\mathbb{P}}^{W}\right)^{\alpha}(\Delta).$$

Proof. This result can be easily proved using Definition 3.9 and Proposition 2.20. □

To conclude, the immediate consequences operator has been generalized in a suitable wat using OWA operators, affording greater tractability in real problems, which is usual the presence of noise (imprecision, uncertainty, etc.).

3.5 Selecting accurate weights vectors to $T_{\mathbb{P}}^{\mathcal{W}}$

It is relevant to take into account that the weights vector is essential for calculate computationally the results and that not every weights vector W is appropriate to generalize the classical immediate consequences operator. For instance, the opposite natural definition of the $T_{\mathbb{P}}$ can be obtained if the minimum OWA, given at Example 2.28, is regarded. With respect to the choice of the weights vector and, consequently, the OWA operator, the most convenient way is to select those that can ensure values close to the supremum, with the aim of not obtaining a definition that considerably differed from the original $T_{\mathbb{P}}$.

Therefore, it is advisable in any new variant of $T_{\mathbb{P}}$ to obtain values close to the greatest one. In order to catalogue the different generalizations of the immediate consequences operator through OWA operators, we will consider the orness measure [15, 32], that can also be seen as the degree of approximation to the highest value given by $@^*$ (Example 2.28).

The first consequence of this measure is that we should consider vectors $W \in [0,1]^m$, with the greatest possible values for the components, that is, $\sum_{i=1}^{m} W[i] = 1$. Moreover, Theorem 2.33 provides a sufficient condition to ensure at least a medium orness value of the considered OWA.

Next, we adapt the orness measure to the general definition of OWA immediate consequences operator.

Definition 3.17. *($\mathcal{W}$-orness). We define the* $\mathcal{W}$-orness measure associated with the family $\mathcal{W}$ *as the mapping*

$$\overline{orness}\colon \Sigma(\mathbb{P}_{A_1}) \times \ldots \times \Sigma(\mathbb{P}_{A_n}) \longrightarrow [0,1],$$

defined as follows

$$\overline{orness}(\mathcal{W}) = \inf\{orness(W_A) \mid W_A \in \mathcal{W}\},$$

where $\Sigma(\mathbb{P}_{A_i}) = \left\{W \in [0,1]^{|\mathbb{P}_{A_i}|} \mid \sum_{j=1}^{|\mathbb{P}_{A_i}|} W[j] \leq 1\right\}$*, for all* $i \in \{1,\ldots,n\}$*, and the orness operator is given by Definition 2.31. Notice that if* $|\mathbb{P}_A| < 2$*, then the aggregator operator is the maximum with orness value of* 1*, which does not affect the computation of the infimum.*

Example 3.18. *Consider Example 2.13 with the extra rule*

$$\langle \texttt{Heat Up} \leftarrow_G \texttt{Sunlight} \wedge_L \texttt{Humidity}, 0.3\rangle\,.$$

As the propositional symbol `Spray Water` *is head of three rules, but* `Heat up` *is head of a single rule, then we should not consider the same weights vector for both rules, but a family of weights vectors, such as*

$$\mathcal{W} = \{W_{\texttt{Spray Water}}, W_{\texttt{Heat Up}}\} = \{W_2, W^*\},$$

where W^* *is the maximum OWA (Example 2.28) and* $W_2 = \left(\frac{1}{2}, \frac{1}{2}, 0\right)$*, which was defined in general in Example 3.5. Consequently, considering the interpretation* I *defined as*

$$\begin{aligned} I(\texttt{Temperature}) &= 0.9, & I(\texttt{Sunlight}) &= 0.8,\\ I(\texttt{Concentration of } CO_2) &= 0.6, & I(\texttt{Humidity}) &= 0.75, \end{aligned}$$

we obtain the following values of the OWA immediate consequences operator.

$$\begin{aligned} T_{\mathbb{P}}^{\mathcal{W}}(I)(\texttt{Spray Water}) &= T_{\mathbb{P}}^{W_{\texttt{Spray Water}}}(I)(\texttt{Spray Water})\\ &= T_{\mathbb{P}}^{W_2}(I)(\texttt{Spray Water})\\ &\overset{(1)}{=} \frac{1}{2}\cdot C_I^{SW}(u_{\pi(1)}) + \frac{1}{2}\cdot C_I^{SW}(u_{\pi(2)}) + 0\cdot C_I^{SW}(u_{\pi(3)})\\ &= \frac{1}{2}\cdot 0.7 + \frac{1}{2}\cdot 0.54 + 0\cdot 0.15\\ &= 0.62\\ T_{\mathbb{P}}^{\mathcal{W}}(I)(\texttt{Heat Up}) &= T_{\mathbb{P}}^{W_{\texttt{Heat Up}}}(I)(\texttt{Heat Up})\\ &= T_{\mathbb{P}}^{W^*}(I)(\texttt{Heat Up})\\ &= 1\cdot C_I^{\texttt{Heat Up}}(u_{\pi(1)})\\ &= 1\cdot 0.3\\ &= 0.3. \end{aligned}$$

where in (1) we have shorten the propositional symbol `Spray Water` *to* `SW`*. From Definition 3.17, we obtain that*

$$\overline{orness}(\mathcal{W}) = \inf\{orness(W_{\texttt{Spray Water}}), orness(W_{\texttt{Heat Up}})\} = \inf\{0.75, 1\} = 0.75.$$

In addition, this example can be used to show Proposition 3.12 and how Definition 3.11 is more appropriate than the first proposal (Definition 3.2). Notice that, if we consider the weights vector $\left(\frac{1}{2}, \frac{1}{2}, 0, \overset{(n-2)}{\ldots}, 0\right)$*, where* n *is the number of rules in* $\mathbb{P}$*, then we obtain that* $T_{\mathbb{P}}^{\tau}(I)(\texttt{Spray Water}) = 0.62$ *and* $T_{\mathbb{P}}^{\tau}(I)(\texttt{Heat Up}) = 0.15$*, where the truth value of* `Heat Up` *is penalized because of only one rule in the program with this propositional symbol as head exists.*

Taking into account Proposition 3.12, there exists the following family of weights vectors

$$\mathcal{W}_1 = \left\{W_{\texttt{Spray Water}} = \left(\frac{1}{2}, \frac{1}{2}, 0\right) \in [0,1]^3, W_{\texttt{Heat Up}} = \frac{1}{2} \in [0,1]\right\},$$

so that $T_{\mathbb{P}}^{\mathcal{W}_1}(I)(\texttt{Spray Water}) = T_{\mathbb{P}}^{\tau}(I)(\texttt{Spray Water})$ *and* $T_{\mathbb{P}}^{\mathcal{W}_1}(I)(\texttt{Heat Up}) = T_{\mathbb{P}}^{\tau}(I)(\texttt{Heat Up})$*.*

4 On the continuity of the $T_{\mathbb{P}}^{\mathcal{W}}$ operator

In this section, we will study conditions to ensure the continuity of the new definition of the immediate consequences operator. As we have already mentioned, we will consider the Scott-continuity definition. This property is essential in order to obtain the least fixed point in numerable (ω) iterations. In the following, we are going to consider a multi-adjoint logic program $\mathbb{P}$, a multi-adjoint Ω-algebra $\mathfrak{L} = \langle L, \mathfrak{H} \rangle$ with respect to the pairs $(\leftarrow_l, \&_l)$, for $l \in \{1, \ldots, m\}$, the complete lattice $\langle [0,1], \leq \rangle$ and the family of weights vectors

$$\mathcal{W} = \left\{ W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1, A \in \Pi, \mathbb{P}_A \neq \varnothing, |\mathbb{P}_A| = n_A \right\},$$

where $n_A \in \mathbb{N}^*$.

Our first result is a generalization of the theorem in the classical framework. This theorem shows that if all operators in $\mathcal{W}$ are Scott-continuous, then $T_{\mathbb{P}}^{\mathcal{W}}$ is Scott-continuous.

Theorem 4.1. *If all the operators occurring in the bodies of the rules of a program $\mathbb{P}$ are Scott-continuous, and the adjoint conjunctions are also Scott-continuous in the second argument, then $T_{\mathbb{P}}^{\mathcal{W}}$ is Scott-continuous.*

Proof. We have to prove, for each directed subset of interpretations X and each atomic formula A, that

$$T_{\mathbb{P}}^{\mathcal{W}}(\sup X)(A) = \sup \left\{ T_{\mathbb{P}}^{\mathcal{W}}(J)(A) \mid J \in X \right\}. \tag{6}$$

We will denote $S = \sup X$. If $\mathbb{P}_A = \varnothing$, then $T_{\mathbb{P}}^{\mathcal{W}}(\sup X)(A) = 0 = \sup \left\{ T_{\mathbb{P}}^{\mathcal{W}}(J)(A) \mid J \in X \right\}$. If $\mathbb{P}_A \neq \varnothing$, then we have the following chain of equalities:

$$\begin{aligned}
T_{\mathbb{P}}^{\mathcal{W}}(\sup X)(A) &= T_{\mathbb{P}}^{W_A}(\sup X)(A) \\
&= \sum_{i=1}^{n_A} W_A[i]\, C_S^A\left(u_{\pi_A(i)}\right) \\
&= \sum_{i=1}^{n_A} W_A[i] \left(\vartheta_i \dot{\&}_i \widehat{S}(\mathcal{B}_i)\right) \\
&\overset{(1)}{=} \sum_{i=1}^{n_A} W_A[i] \left(\vartheta_i \dot{\&}_i \sup \left\{ \hat{J}(\mathcal{B}_i) \mid J \in X \right\}\right) \\
&\overset{(2)}{=} \sum_{i=1}^{n_A} W_A[i] \sup \left\{ \vartheta_i \dot{\&}_i \hat{J}(\mathcal{B}_i) \mid J \in X \right\} \\
&\overset{(3)}{=} \sup \left\{ \sum_{i=1}^{n_A} W_A[i] \left(\vartheta_i \dot{\&}_i \hat{J}(\mathcal{B}_i)\right) \mid J \in X \right\} \\
&= \sup \left\{ \sum_{i=1}^{n_A} W_A[i]\, C_J^A\left(u_{\pi(i)}\right) \mid J \in X \right\} \\
&= \sup \left\{ T_{\mathbb{P}}^{W_A}(J)(A) \mid J \in X \right\} \\
&= \sup \left\{ T_{\mathbb{P}}^{\mathcal{W}}(J)(A) \mid J \in X \right\},
\end{aligned}$$

where equality (1) follows from Lemma 2.26, equality (2) follows from the Scott-continuity of the operators $\&_l$ on the second argument and equality (3) follows from the fact that the product and the sum are continuous mappings and so, Scott-continuous on the unit interval, in particular.

Therefore, we can conclude that the new definition of the immediate consequences operator, $T_{\mathbb{P}}^{\mathcal{W}}$, is Scott-continuous. □

In the following example, we will show that the consideration of directed sets cannot be dropped from the definition of continuity, in order to satisfy Equation 6.

Example 4.2. *Consider the multi-adjoint Ω-algebra $\langle [0,1], \leq, \&_1, \&_2\rangle$ and the program $\mathbb{P}$ composed of the rules $\langle A \leftarrow_1 B, \vartheta_1\rangle$ and $\langle A \leftarrow_2 B \wedge C, \vartheta_2\rangle$, with $0 \neq \vartheta_2 < \vartheta_1$. Let X be the set of interpretations $\{I_1, I_2\}$, where I_1 and I_2 are defined as follows:*

$$\begin{array}{lll} I_1(A) = 0, & I_1(B) = 1 & I_1(C) = 0, \\ I_2(A) = 0, & I_2(B) = 0 & I_2(C) = 1. \end{array}$$

Clearly X is not directed, because $\{I_1, I_2\} \subseteq X$ has not an upper bound in X. Suppose that we are considering the weights vector $W_1 = (0, 1, 0, \ldots, 0)$. Then, it is verified that

$$\begin{array}{rcl} \sup\left\{T_{\mathbb{P}}^{W_1}(J)(A) \mid J \in X\right\} & = & \sup\left\{T_{\mathbb{P}}^{W_1}(I_1)(A), T_{\mathbb{P}}^{W_1}(I_2)(A)\right\} \\ & = & \sup\{0, 0\} \\ & = & 0. \end{array}$$

On the other hand, we can write $\sup X = S$ and it is defined as follows.

$$S(A) = 0, \qquad S(B) = 1, \qquad S(C) = 1.$$

Then, we have the following chain of equalities:

$$\begin{array}{rcl} T_{\mathbb{P}}^{W_1}\left(\sup X\right)(A) & = & \displaystyle\sum_{i=1}^{2} W_i C_S^A(u_{\pi(i)}) \\ & = & \vartheta_2, \end{array}$$

because of $\vartheta_1 \dot{\&}_1 \hat{S}(B) = \vartheta_1 \dot{\&}_1 1 = \vartheta_1$, $\vartheta_2 \dot{\&}_2 \hat{S}\left(B \wedge C\right) = \vartheta_2 \dot{\&}_2 1 = \vartheta_2$, and that W_1 consider the second greatest element in the set. Consequently, as $0 \neq \vartheta_2, \vartheta_1$, Equality 6 does not hold for arbitrary subsets $X \subseteq \mathcal{I}_{\mathfrak{L}}$.

We present the following result in which we prove that the Scott-continuous of the adjoint conjuntor is a necessary condition for the Scott-continuous of the operator $T_{\mathbb{P}}^{\mathcal{W}}$.

Theorem 4.3. *If the operator $T_{\mathbb{P}}^{\mathcal{W}}$ is Scott-continuous for all program $\mathbb{P}$ in $\mathfrak{L}$, then any operator in the body of the rules is Scott-continuous and the adjoint conjunctors $\&_l$, with $l \in \{1, \ldots, m\}$, are Scott-continuous in the second argument.*

Proof. First of all, we start with the Scott-continuity of the adjoint operator $\&_l$ in the second argument, with $l \in \{1, \ldots, m\}$. Given $l \in \{1, \ldots, m\}$, $\vartheta \in L$ and the directed set $Y \subseteq L$, we are going to prove

$$\sup\{\vartheta \&_l y \mid y \in Y\} = \vartheta \&_l \sup Y.$$

Given a propositional symbol $B \in \Pi$ and the set of interpretations $X_Y = \{I_y \colon \Pi \to [0,1] \mid y \in Y\}$ such that

$$I_y(C) = \begin{cases} y & \text{if} \quad C = B \\ 0 & \text{otherwise} \end{cases}$$

for all $C \in \Pi$. We are going to prove that X_Y is directed. Given a subset $\{I_{y_1}, \ldots, I_{y_k}\}$ of X_Y, where $\{y_1, \ldots, y_k\} \subseteq Y$, as Y is a directed set, then there exists $y_M \in Y$ such that it is an upper bound of $\{y_1, \ldots, y_k\}$. Therefore, the interpretation I_{y_M} is an upper bound of $\{I_{y_1}, \ldots, I_{y_k}\}$. Hence, X is directed.

Consider the program $\mathbb{P}$ in $\mathfrak{L}$ composed of the rule: $\mathbb{P} = \{\langle A \leftarrow_l B, \vartheta\rangle\}$. Then, we have the following chain of equalities:

$$\begin{array}{rcl} \sup\left\{\vartheta \dot{\&}_l y \mid y \in Y\right\} & = & \sup\left\{\vartheta \dot{\&}_l I_y(B) \mid I_y \in X_Y\right\} \\ & \overset{(1)}{=} & \sup\left\{\displaystyle\sum_{i=1}^{n_A} W_A[i]\left(\vartheta_i \dot{\&}_i I_y\left(B_i\right)\right) \mid I_y \in X_Y\right\} \\ & = & \sup\left\{T_{\mathbb{P}}^{W_A}(I_y)(A) \mid I_y \in X_Y\right\} \\ & \overset{(2)}{=} & T_{\mathbb{P}}^{W_A}\left(\sup X_Y\right)(A) \\ & = & \displaystyle\sum_{i=1}^{n_A} W_A[i]\left(\vartheta_i \dot{\&}_i \widehat{\sup X_Y}(B_i)\right) \\ & \overset{(3)}{=} & \vartheta \dot{\&}_l \sup Y, \end{array}$$

where (1) is verified taking into account that $I_y(C) = 0$, for all $C \in \Pi$, with $C \neq B$. Equality (2) follows from the Scott-continuity of $T_{\mathbb{P}}^{\mathcal{W}}$. Finally, (1) follows from the fact that $\sup X_Y(C) = \sup Y$, if $C = B$, and it is 0, otherwise. Therefore, $\&_l$ is continuous in the second argument, for all $l \in \{1, \ldots, m\}$.

Now, we are going to prove the continuity of the operators in the body of the rules. In order to simplify the process we will use the following notation. If $\bar{y}$, $\bar{z} \in L^n$, where $\bar{y} = (y_1, \ldots, y_n)$, $\bar{z} = (z_1, \ldots, z_n)$, then $\bar{y} \leq \bar{z}$ if and only if $y_i \leq z_i$ for all $i \in \{1, \ldots, n\}$.

Given a directed set $Y \subseteq L^n$, we have to prove that

$$\dot{@}\left(\sup Y\right) = \sup\left\{\dot{@}\left(y_1, \ldots, y_n\right) \mid (y_1, \ldots, y_n) \in Y\right\},$$

where @ is an operator (aggregator) in the body of one of the rules in the program. Clearly, if $(y_1, \ldots, y_n) \in Y$, then $(y_1, \ldots, y_n) \leq \sup Y$ and by the monotonocity of $\dot{@}$, $\dot{@}\left(y_1, \ldots, y_n\right) \preceq \dot{@}\left(\sup Y\right)$. Therefore,

$$\sup\left\{\dot{@}\left(y_1, \ldots, y_n\right) \mid (y_1, \ldots, y_n) \in Y\right\} \leq \dot{@}\left(\sup Y\right),$$

For the other inequality, given n propositional symbols $A_1, \ldots, A_n \in \Pi$ and $\bar{y} = (y_1, \ldots, y_n) \in Y$, we can consider the interpretation $I_{\bar{y}} \colon \Pi \to [0, 1]$ defined as

$$I_{\bar{y}}(C) = \begin{cases} y_i & \text{if} \quad C = A_i \\ 0 & \text{otherwise} \end{cases}$$

for all $C \in \Pi$. On the one hand, we can consider the set $X_Y = \{I_{\bar{y}} \in \mathcal{I}_{\mathfrak{L}} \mid \bar{y} \in Y\}$ and its supremum $S_Y = \sup X_Y$. From the definition of $I_{\bar{y}}$ and as Y is directed, then X_Y is also directed. For all $\bar{y} \in Y$ it is verified that

$$(y_1, \ldots, y_n) = (I_{\bar{y}}(A_1), \ldots, I_{\bar{y}}(A_n)) \leq (S_Y(A_1), \ldots, S_Y(A_n)).$$

Hence, it is satisfied that

$$\sup Y \leq (S_Y(A_1), \ldots, S_Y(A_n)).$$

As a consequence, by the monotonocity of $\dot{@}$, we reach

$$\dot{@}\left(\sup Y\right) \leq \dot{@}\left(S_Y(A_1), \ldots, S_Y(A_n)\right) = \widehat{S_Y}\left(@\left(A_1, \ldots, A_n\right)\right).$$

On the other hand, consider the multi-adjoint logic program $\mathbb{P}$ composed of the rule: $u = \langle A \leftarrow_l @\left(A_1, \ldots, A_n\right), \top\rangle$. We have the following chain of equalities.

$$\begin{aligned}
\widehat{S_Y}\left(@\left(A_1, \ldots, A_n\right)\right) &= \top \dot{\&}_l \widehat{S_Y}\left(@\left(A_1, \ldots, A_n\right)\right) \\
&\overset{(1)}{=} \sum_{i=1}^{n_A} W_A[i] C_{S_Y}^A\left(u_{\pi(i)}\right) \\
&= T_{\mathbb{P}}^{W_A}\left(S_Y\right)(A) \\
&\overset{(2)}{=} \sup\left\{T_{\mathbb{P}}^{W_A}\left(J_{\bar{y}}\right)(A) \mid J_{\bar{y}} \in X_Y\right\} \\
&= \sup\left\{\sum_{i=1}^{n_A} W_A[i] C_{J_{\bar{y}}}^A(u_{\pi(i)}) \mid J_{\bar{y}} \in X_Y\right\} \\
&= \sup\left\{C_{J_{\bar{y}}}^A(u) \mid J_{\bar{y}} \in X_Y\right\} \\
&= \sup\left\{\top \dot{\&}_l \widehat{J_{\bar{y}}}\left(@\left(A_1, \ldots, A_n\right)\right) \mid J_{\bar{y}} \in X_Y\right\} \\
&\overset{(3)}{=} \sup\left\{\dot{@}\left(y_1, \ldots, y_n\right) \mid (y_1, \ldots, y_n) \in Y\right\}.
\end{aligned}$$

where (1) and (3) follow from the definition of $I_{\bar{y}}$ and (2) follows from the Scott-continuity of $T_{\mathbb{P}}^{W_A}$. Consequently, it is verified that

$$\begin{aligned}
\sup\left\{\dot{@}\left(y_1, \ldots, y_n\right) \mid (y_1, \ldots, y_n) \in Y\right\} &\leq \dot{@}\left(\sup Y\right) \\
&\leq \widehat{S_Y}\left(@\left(A_1, \ldots, A_n\right)\right) \\
&= \sup\left\{\dot{@}\left(y_1, \ldots, y_n\right) \mid (y_1, \ldots, y_n) \in Y\right\}.
\end{aligned}$$

□

Finally, from the previous results, we obtain the following version of Proposition 2.24 to the new more flexible immediate consequences operator.

Corollary 4.4. *If $T_{\mathbb{P}}^{\mathcal{W}}$ is continuous, then $\left(T_{\mathbb{P}}^{\mathcal{W}}\right)^{\omega}(\Delta)$ is the least model of $\mathbb{P}$, that is, $\left(T_{\mathbb{P}}^{\mathcal{W}}\right)^{\omega}(\Delta) = M_{\mathbb{P}}$.*

Proof. The proof follows from Proposition 2.24 and the previous results. □

5 Conclusions and future work

Following the philosophy and motivation already introduced in [22, 23], this paper has presented a flexible definition of the immediate consequences operator based on the use of OWA operators, which has allowed us to effectively solve the problem of the possible loss of information caused by the use of the supremum. The semantics of a multi-adjoint logic program have also been conveniently adapted to this new framework through the use of a family of weights vectors. In addition, this new operator verifies very interesting properties such as the monotonicity and the continuity.

However, some open questions remain. The first one refers to the possible relationship that can be established between generalized quantifiers and OWA operators when the immediate consequences operator is to be generalized. Not all fuzzy measures (in the case of generalized quantifiers) nor all weights vectors (in the case of OWA operators) serve to generalize the $T_{\mathbb{P}}$. Consequently, it is interesting to study conditions that make it possible to ensure that a certain generalization of the $T_{\mathbb{P}}$ is adequate. Moreover, we will study the possibility of considering other kind of general operators [5, 12, 24].

On the other hand, all the results presented in this paper have been performed on the unit interval, which is a complete lattice that is totally ordered. New generalizations of the immediate consequences operator on a general lattice will be studied as future work.

Acknowledgement

Partially supported by the 2014-2020 ERDF Operational Programme in collaboration with the State Research Agency (AEI) in project PID2019-108991GB-I00, and with the Department of Economy, Knowledge, Business and University of the Regional Government of Andalusia in project FEDER-UCA18-108612, and by the European Cooperation in Science and Technology (COST) Action CA17124.

References

[1] E. Bas, U. Yolcu, E. Egrioglu, *Picture fuzzy regression functions approach for financial time series based on ridge regression and genetic algorithm*, Journal of Computational and Applied Mathematics, **370** (2020), 112656.

[2] A. Colmerauer, H. Kanoui, P. Roussel, R. Pasero, *Un systeme de communication homme-machine en Francais*, Technical report, Groupe de Recherche en Intelligence Artificielle, Universidad de Aix-Marseille, 1973.

[3] M. E. Cornejo, D. Lobo, J. Medina, *On the solvability of bipolar max-product fuzzy relation equations with the product negation*, Journal of Computational and Applied Mathematics, **354** (2019), 520-532.

[4] M. E. Cornejo, J. Medina, E. Ramrez-Poussa, *Algebraic structure and characterization of adjoint triples*, Fuzzy Sets and Systems, **425** (2021), 117-139.

[5] M. E. Cornejo, J. Medina, I. Stajner-Papuga, A. Tepavcevic, *Lattice valued aggregation operators applicable in digital forensics*, Studies in Computational Intelligence, (2023), 1-7.

[6] C. V. Damásio, L. M. Pereira, *Monotonic and residuated logic programs*, Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01, pages 748-759. Lecture Notes in Artificial Intelligence, 2143, 2001.

[7] M. Davis, H. Putnam, *A computing procedure for quantification theory*, Journal of the ACM, **7** (1960), 201-215.

[8] M. V. Emden, R. Kowalski, *The semantics of predicate logic as a programming language*, Journal of the ACM, **23**(4) (1976), 733-742.

[9] J. H. Gallier, *Logic for computer science: Foundations of automatic theorem proving*, Harper and Row Publishers, Inc., USA, 1985.

[10] P. Gilmore, *A proof method for quantification theory*, IBM Journal of Research and Development, **4** (1960), 28-35.

[11] N. V. Hung, V. M. Tam, N. H. Tuan, D. O'Regan, *Convergence analysis of solution sets for fuzzy optimization problems*, Journal of Computational and Applied Mathematics, **369** (2020), 112615.

[12] D. Jocic, I. Stajner-Papuga, *Distributivity between 2-uninorms and Mayor's aggregation operators*, Iranian Journal of Fuzzy Systems, **19**(6) (2022), 13-27.

[13] P. Julián-Iranzo, C. Rubio-Manzano, *A sound and complete semantics for a similarity-based logic programming language*, Fuzzy Sets and Systems, **317** (2017), 1-26.

[14] M. Kifer, V. S. Subrahmanian, *Theory of generalized annotated logic programming and its applications*, Journal of Logic Programming, **12** (1992), 335-367.

[15] A. Kishor, A. K. Singh, N. R. Pal, *Orness measure of OWA operators: A new approach*, IEEE Transactions on Fuzzy Systems, **22**(4) (2014), 1039-1045.

[16] R. Kowalski, *Predicate logic as a programming language*, Proceedings of IFIP'74, North Holland. Publishing Company, (1974), 569-574.

[17] I. Lizasoain, C. Moreno, *OWA operators defined on complete lattices*, Fuzzy Sets and Systems, **224** (2013), 36-52.

[18] J. Lloyd, *Foundations of logic programming*, Springer Verlag, 1987.

[19] J. Medina, M. Ojeda-Aciego, P. Vojtáš, *Multi-adjoint logic programming with continuous semantics*, Lecture Notes in Artificial Intelligence, **2173** (2001), 351-364.

[20] J. Medina, M. Ojeda-Aciego, P. Vojtáš, *Similarity-based unification: A multi-adjoint approach*, Proc. EUSFLAT Conference in Fuzzy Logic and Technology, (2001), 273-276.

[21] J. Medina, M. Ojeda-Aciego, P. Vojtáš, *Similarity-based unification: A multi-adjoint approach*, Fuzzy Sets and Systems, **146** (2004), 43-62.

[22] J. Medina, J. A. Torné-Zambrano, *Fuzzy logic programming with generalized quantifiers*, Springer International Publishing, **1040** (2023), 17-23.

[23] J. Medina, J. A. Torné-Zambrano, *Immediate consequences operator on generalized quantifiers*, Fuzzy Sets and Systems, **456** (2022), 72-91.

[24] R. Mesiar, A. Kolesarova, S. Borkotokey, *2-set-based extended functions*, Iranian Journal of Fuzzy Systems, **18**(3) (2021), 13-26.

[25] M. Ojeda-Hernndez, I. P. Cabrera, P. Cordero, *Quasi-closed elements in fuzzy posets*, Journal of Computational and Applied Mathematics, **404** (2022), 113390.

[26] D. Prawitz, *An improved proof procedure*, Theoria, **26** (1960), 102-139.

[27] U. Straccia, M. Ojeda-Aciego, C. V. Damásio, *On fixed-points of multivalued functions on complete lattices and their application to generalized logic programs*, SIAM Journal on Computing, **38**(5) (2009), 1881-1911.

[28] A. Tarski, *A lattice-theoretical fixpoint theorem and its applications*, Pacific Journal of Mathematics, **5**(2) (1955), 285-309.

[29] M. H. Van Emden, *Quantitative deduction and its fixpoint theory*, The Journal of Logic Programming, **1** (1986), 37-53.

[30] P. Vojtáš, *Fuzzy logic programming*, Fuzzy Sets and Systems, **124**(3) (2001), 361-370.

[31] R. R. Yager, *On ordered weighted averaging aggregation operators in multi-criteria decision making*, IEEE Transactions on Systems, Man, and Cybernetics, **18**(1) (1988), 183-190.

[32] R. R. Yager, *Families of OWA operators*, Fuzzy Sets and Systems, **59**(2) (1993), 125-148.

Capítulo 7

Semántica operacional sobre programación lógica de primer orden, difusa y arbitraria

Operational semantics on arbitrary first-order fuzzy logic programming

Jesús Medina[a], José Antonio Torné-Zambrano[a]

[a]*Department of Mathematics, University of Cádiz, Cádiz, Spain*

Abstract

Intelligent systems require the development of trustworthy automatic procedures providing recommendations, fault detections, predictive behaviours, etc. Fuzzy logic, in general, and fuzzy logic programming, in particular, are robust and traceable formal tools to model real data sets, which usually contain imperfect and uncertain data. This paper introduces a procedural semantics that can be used on any monotonic version of the immediate consequences operator defined in a first-order fuzzy logic programming framework.

Keywords: Fuzzy logic programming, immediate consequences operator, procedural semantics, fixed-point semantics.

1. Introduction

Logic programming is a branch of computational logic whose main objective is to formally represent knowledge and problems in a more natural way for the human intellect, and to obtain automatic conclusions from the formal system, through an inference mechanism [19, 9, 17]. It was born from the efforts of different authors in the field of automatic deduction and artificial intelligence [34, 11, 33, 10, 8]. This new theory was able to take advantage of the greater expressiveness provided by the introduction

*Partially supported by the project PID2022-137620NB-I00 funded by MICIU/AEI/10.13039/501100011033 and FEDER, UE, by the grant TED2021-129748B-I00 funded by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR.

Email addresses: jesus.medina@uca.es (Jesús Medina), joseantonio.torne@uca.es (José Antonio Torné-Zambrano)

of fuzzy sets [43], giving rise to new programming paradigms, such the fuzzy logic programming, which incorporates uncertainty and imprecision in problem-solving scenarios [40]. Among all the generalizations of fuzzy logic programming, one of the most interesting frameworks is the multi-adjoint one, which provides greater flexibility than other approaches by allowing the use of a wide variety of adjoint pairs in the construction of program rules, as well as less restrictive properties in the considered connectives [21, 6, 24]. From a real-world perspective, the theoretical framework of fuzzy logic programming has been implemented through systems like FASILL [14] or Bousi-Prolog [13, 15], and have been applied in several areas such as social networks [3], knowledge discovery [36], explainable machine learning [37], decision making [5], cloud computing [39], genome-scale metabolic models [20], neural networks [30], SAT/SMT techniques [2], semantic web [1] and text classification [35].

From a syntactic point of view, the main notion of multi-adjoint logic programming is that of logic program, which is a set of rules and facts that model a certain system of knowledge. On the other hand, semantically there are different ways of giving meaning to a logic program. In [21], a fixed-point semantics was studied, which is based on the notion of immediate consequences operator $T_{\mathbb{P}}$, an essential operator for deriving conclusions and deductions from databases modeled through logic rules. The fixed-point semantics provides a method for obtaining the main consequences of a logic program through the least fixed-point of the immediate consequences operator. The classical definition of the $T_{\mathbb{P}}$, provided by van Emden and Kowalski in [9], is defined as the supremum of the values provided by each logic rule. Nevertheless, the inherent strictness of the supremum can cause that a bad quality of the data alters the final result of the computation, as it can be seen, from a didactic point of view, in [26, Example 5]. Different generalizations have been studied with the aim of addressing this problem and making the operator more flexible. In [26], a new definition based on generalized quantifiers was proposed, whereas in [27], OWA operators were used for this purpose. These approaches allow for a more versatile final outcome in the computation, based not just on the requirements of users, but also on the possible presence of noise in the data and the need to adapt to this framework. Once a fixed-point semantics has been studied [26, 27], for the development of a computational model it is necessary to specify an operational semantics [22], which allows to define an automatic reasoning procedure for obtaining deductions and informations from

the modeled system, and for providing answers to queries to the system. Recently, a procedural semantics has been introduced in the framework of propositional multi-adjoint logic programming with generalized quantifiers in [28].

In this paper, the first-order multi-adjoint logic programming framework [12, 24] will be taken into account and, instead of considering a particular extension of the $T_{\mathbb{P}}$, a general immediate consequences operator $T_{\mathbb{P}}^{\diamond}$ will be considered, which naturally will be monotonic in the first-order logic framework. Furthermore, an operational semantics will be needed when we are considering a new and more flexible computational model based on the $T_{\mathbb{P}}^{\diamond}$ operator, in order to computationally construct a precise algorithm for obtaining correct and computed answers for a given query.

Therefore, for the operator $T_{\mathbb{P}}^{\diamond}$, an operational semantics will be defined in general, allowing to be applied to every monotonic immediate consequences operator $T_{\mathbb{P}}^{\diamond}$, such as the one introduced in [12] and the first order extension of the versions given in [26, 27]. As a consequence, the results in [28, 12] will be extended in this new approach.

The paper is structured as follows. Section 2 will recall some preliminary results; Section 3 will present the general monotonic immediate consequences operator, together with the particular case of the OWA immediate consequences operator, which was introduced in [27]. In Section 4 an operational semantics in this new flexible framework will be studied, including some correctness results, and a new definition of reductants based on the immediate consequences operator will be given to solve the problem of not being able to computationally calculate the correct answer that provides the most significant information. In Section 5 the soundness of the new procedure will be studied, whereas in Section 6 completeness results will be analyzed. Finally, some conclusions and future work will be presented in Section 7.

2. Preliminaries

We will begin recalling some relevant definitions that will be necessary in the next sections [21, 23].

2.1. Multi-adjoint algebra structure

First of all, it is imperative to introduce the concept of *adjoint pair*, as a basic element of the algebraic structures that will be considered. Adjoint

pairs were introduced by Jan Pavelka in a logical framework in [32]. Next, we recall a general definition considered in [21, 7].

Definition 1 ([21]). Let $\langle P, \preceq \rangle$ be a poset and $(\leftarrow, \&)$ be a pair of binary operators in P such that

1. The operator $\&$ is increasing in both arguments.
2. The operator $\leftarrow$ is increasing in the first argument (*consequent*) and decreasing in the second one (*antecedent*)
3. The adjoint property is satisfied, i.e., $x \preceq (y \leftarrow z)$ holds if and only if $(x \& z) \preceq y$ holds, for all $x, y, z \in P$.

Then, we say that $(\leftarrow, \&)$ is an *adjoint pair* in $\langle P, \preceq \rangle$.

The algebraic structure that makes possible the use of different adjoint pairs in a complete lattice is recalled below.

Definition 2 ([23]). Let $\langle L, \preceq \rangle$ be a complete lattice. A *multi-adjoint lattice* $\mathcal{L}$ is a tuple $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ satisfying that

1. $\langle L, \preceq \rangle$ is bounded, i.e., it has a bottom $\bot$ and a top $\top$ element.
2. $(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq \rangle$, for all $i \in \{1, \ldots, n\}$.
3. $\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$, for all $\vartheta \in L$ and $i \in \{1, \ldots, n\}$.

In order to represent the language in a natural way, the possibility of using additional operators along with those of the multi-adjoint lattice is suggestive. For this reason the following algebraic structure is introduced, where the definitions of graded set, Ω-algebra and Ω-subalgebra can be consulted in [21, 23].

Definition 3 ([23]). Let Ω be a graded set containing operators $\leftarrow_i$ and $\&_i$, for $i \in \{1, \ldots, n\}$, and possibly some extra operators, and $\mathfrak{L} = \langle L, I \rangle$ be an Ω-algebra whose carrier set L is a complete lattice under $\preceq$. We say that $\mathfrak{L}$ is a *multi-adjoint Ω-algebra* with respect to the pairs $(\leftarrow_i, \&_i)$, for $i \in \{1, \ldots, n\}$, if $\mathcal{L} = (L, \preceq, I(\leftarrow_1), I(\&_1), \ldots, I(\leftarrow_n), I(\&_n))$ is a multi-adjoint lattice.

Now, an example of multi-adjoint Ω-algebra is introduced. As a simplification, we will use the dot on the symbol $\&$ for denoting the interpretation of the operator, $I(\&)$.

Example 1. Consider the graded set $\Omega = \left\{\leftarrow_{\mathrm{P}}, \&_{\mathrm{P}}, \leftarrow_{\mathrm{G}}, \&_{\mathrm{G}}, \wedge_{\mathrm{Ł}}, @_{(2,1)}\right\}$, together with an Ω-algebra $\mathfrak{L} = \langle [0,1], I\rangle$, where $\dot{\&}_{\mathrm{P}}$ and $\dot{\&}_{\mathrm{G}}$ are the product and Gödel t-norms respectively, along with their corresponding residuated implications, and with the additional operator, the Łukasiewicz conjunctor $\dot{\wedge}_{\mathrm{Ł}}$. As $([0,1], \leq, \dot{\leftarrow}_{\mathrm{P}}, \dot{\&}_{\mathrm{P}}, \dot{\leftarrow}_{\mathrm{G}}, \dot{\&}_{\mathrm{G}})$ is a multi-adjoint lattice, then $\mathfrak{L}$ is a multi-adjoint Ω-algebra. As a reminder, these operators are defined, for all x, y, $z \in [0,1]$, as

$$\dot{\&}_{\mathrm{G}}(x,y) = \min\{x,y\} \qquad z\dot{\leftarrow}_{\mathrm{G}}y = \begin{cases} 1 & \text{if} \quad y \leq z \\ z & \text{otherwise} \end{cases}$$

$$\dot{\&}_{\mathrm{P}}(x,y) = x \cdot y \qquad z\dot{\leftarrow}_{\mathrm{P}}y = \begin{cases} 1 & \text{if} \quad y \leq z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$

$$\dot{\wedge}_{\mathrm{Ł}}(x,y) = \max\{0, x+y-1\} \qquad \dot{@}_{(2,1)}(x,y) = \frac{2x+y}{3}$$

□

Finally, we recall the definition of (weak) aggregator, for the new definition of reductant.

Definition 4. Let $\langle L, \preceq\rangle$ be a complete lattice. A mapping $@\colon L^n \to L$ is said to be a *weak aggregator* if

1. $@$ satisfies the boundary condition $@(\bot, \ldots, \bot) = \bot$.
2. $@$ is increasing, i.e., for each $i \in \{1, \ldots, n\}$, if $x_i \preceq y$, then

$$@(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \preceq @(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$$

for all $y, x_1, \ldots, x_n \in L$.

In addition, we say that $@$ is an *aggregator* if it is a weak aggregator and it satisfies that $@(\top, \ldots, \top) = \top$.

2.2. Syntax of multi-adjoint logic programming

Once the algebraic structure that is going to be used for the construction of the logic programs has been analyzed, we will recall some aspects related to the syntax of multi-adjoint logic programming. For this, we need to have a set of symbols (*alphabet*) and a series of syntactic rules (*expressions*) with which to form well-formed formulas. For a more in-depth analysis see [4].

Definition 5. Let Ω be a graded set, $\mathcal{V}$ be a countable infinite set of variables, $\mathcal{P}$ be a non-empty countable set of symbols of predicates, $\mathcal{F}$ be a countable set of symbols of functions and $S = \{\text{“(”}, \text{“)”}, \text{“,”}, \text{“}\forall\text{”}, \text{“}\exists\text{”}\}$. The *alphabet* $\mathfrak{A}$ of multi-adjoint logic programming is the disjoint union $\Omega \uplus \mathcal{V} \uplus \mathcal{P} \uplus \mathcal{F} \uplus S$. In addition, the set of strings of the alphabet will be denoted as $\mathfrak{A}^*$.

The strings of symbols that represent elements of the universe of discourse are *terms*.

Definition 6. The set of *terms* from $\mathfrak{A}$ is the inductive closure of the set $\mathcal{V} \cup \Omega_0$ from the function symbols set $\mathcal{F}$, where $\Omega_0 \subseteq \Omega$ is the subset of constant symbols of the language. The set of terms will be denoted as $\mathcal{T}\text{erm}$.

Predicates are applied to the terms in order to obtain *atoms*, specifying the relationship or property between the terms.

Definition 7. *Atoms or atomic formulae* are the elements of the set of strings of the alphabet, $\mathfrak{A}^*$, of the form $P(t_1, \ldots, t_n)$, where P is an n-ary predicate symbol in $\mathcal{P}$ and $t_1, \ldots, t_n \in \mathcal{T}\text{erm}$. We will denote the set of atoms by $\mathcal{A}\text{tom}$. The particular case of an atom without variables will be called *ground atom*.

Not all expressions have a syntactic meaning. For this, we have the following definition in which we have included the truth values as part of the language.

Definition 8. Given a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$ and a truth value set L, the set of *well-formed formulas*, denoted as $\mathfrak{F}$, is the inductive closure of $\mathcal{A}\text{tom} \cup L$ from the symbols

$$\{\omega \in \Omega_n \mid 1 \leq n\} \cup \{U_x, E_x \mid x \in \mathcal{V}\}$$

where $U_x, E_x \colon \mathfrak{A}^* \to \mathfrak{A}^*$ are defined as $U_x(A) = (\forall x)A$ and $E_x(A) = (\exists x)A$, for all $A \in \mathfrak{A}^*$, and $\Omega_n \subseteq \Omega$ is the subset of symbols of the graded set with arity n.

This set associated with the alphabet, composed of a graded set, variables and symbol of functions and predicates, comprises the *first-order language (of a multi-adjoint logic programming)*, which will be denoted as $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$, in short L_1.

Given a formula $A \in \mathfrak{F}$, we will define the following sets for the sake of readability.

$$\begin{aligned} \mathcal{S}_g(A) &= \{\xi \mid \xi \text{ is a substitution such that } A\xi \text{ is ground}\} \\ \mathcal{V}ar_l(A) &= \{x \mid x \text{ is a free variable of } A\} \end{aligned} \tag{1}$$

When there is no doubt about the formula A we are referring to, the corresponding sets will be denoted as $\mathcal{S}_g$ and $\mathcal{V}ar_l$. From now on, we will fix a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$ and a multi-adjoint Ω-algebra $\mathfrak{L} = \langle L, I\rangle$. The main concept of the syntax of multi-adjoint logic programming is that of *multi-adjoint logic program*, MALP in short, which is usually defined as a set of rules and facts, with the following syntactic conditions.

Definition 9 ([12]). A *multi-adjoint logic program* (MALP) is a set $\mathbb{P}$ of weighted rules of the form $\langle H \leftarrow_i \mathcal{B}; \vartheta\rangle$, also denoted $H \overset{\vartheta}{\leftarrow}_i \mathcal{B}$, such that

1. The rule $\langle H \leftarrow_i \mathcal{B}; \vartheta\rangle$ is a formula of $\mathfrak{F}$, where H is an atomic formula, usually called the *head*; $\mathcal{B}$ is a formula called the *body*; and $\vartheta \in L$ is a truth-value.
2. The body formula $\mathcal{B}$ is built from atomic formulas $B_1, \ldots, B_n$, $0 \leq n$, truth values of L and aggregation operators, by the use of conjunctors $\&_1, \ldots, \&_m$ and $\wedge_1, \ldots, \wedge_k$, disjunctors $\vee_1, \ldots, \vee_{\text{Ł}}$ and aggregators in general $@_1, \ldots, @_r$.
3. Rules whose body is $\top$ are called *facts*.
4. Free occurrences of variables in the program are assumed to be universally quantified.

Note that, the truth value set has the structure of a complete lattice $(L, \preceq)$ and $\top$ is its top element. Moreover, we should also consider that the body $\mathcal{B}$ can alternatively be expressed as $@\,[B_1, \ldots, B_n]$, where $@$ is the composition of the symbols of operators in $\mathcal{B}$ and $B_1 \ldots, B_n$ are the atoms that appear in $\mathcal{B}$. We will finish this section with an example of a logic program.

Example 2. Let $\mathfrak{L}$ be the multi-adjoint Ω-algebra already presented in Example 1, where each operator symbol in Ω is related to their respective interpretations. Under specific conditions, the following logic program $\mathbb{P}$ can be considered.

$$\begin{aligned} \mathcal{R}_1\colon &\quad p(a, Y) \overset{0.8}{\leftarrow}_{\text{G}} && @_{(2,1)}\,(q(X) \&_{\text{G}} r(Y), s(X, b)) \\ \mathcal{R}_2\colon &\quad p(X, b) \overset{0.9}{\leftarrow}_{\text{P}} && t(X) \wedge_{\text{Ł}} r(Y) \\ \mathcal{R}_3\colon &\quad q(X) \overset{0.2}{\leftarrow}_{\text{P}} && t(X) \end{aligned}$$

In each specific case, various values of different variables (hypotheses) are measured and formally expressed as facts, which complete the definition of the program under consideration.

$$\begin{array}{lll} \mathcal{R}_4\colon & r(b) \overset{0.6}{\leftarrow} & \top \\ \mathcal{R}_5\colon & s(a,b) \overset{0.8}{\leftarrow} & \top \\ \mathcal{R}_6\colon & t(a) \overset{0.6}{\leftarrow} & \top \end{array}$$

□

2.3. Semantics of multi-adjoint logic programming

This section introduces the semantics of the logical framework under consideration, which will allow us to establish a deduction system to obtain which consequences are correct. The first definition presents the notion of interpretation of a program $\mathbb{P}$. From now on, a MALP $\mathbb{P}$ will be fixed.

Definition 10 ([24]). An *interpretation* is a mapping from the Herbrand base, $B_{\mathbb{P}}$, of the program $\mathbb{P}$ to the multi-adjoint lattice of truth-values $\langle L, \preceq \rangle$. The set of all interpretations with respect to the Ω-algebra $\mathfrak{L}$ is denoted as $\mathcal{I}_{\mathfrak{L}}$.

Notice that we can naturally extend each interpretation I to be defined on the set of all ground formulas of the language, using the homomorphic extension theorem [21]. This extension will be denoted by $\hat{I}$. The extension for a non-ground formula A is as follows.

$$\hat{I}(A) = \inf\{\hat{I}(A\xi) \mid A\xi \text{ is a ground instantiation of } A\} \tag{2}$$

In addition, the ordering $\preceq$ of $\langle L, \preceq \rangle$ can also be extended to a point-wise ordering in $\mathcal{I}_{\mathfrak{L}}$, forming a complete lattice, denoted by $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$ [24]. The least interpretation $\triangle$ assigns to every ground atomic formula the least element $\bot$ of L. From Expression (2), it can be straightforwardly obtained the following technical lemma.

Lemma 1 ([12]). *Let A be a formula, I be an interpretation and θ be a substitution. Then, $I(\mathcal{A}) \leq I(\mathcal{A}\theta)$.*

Now, the definition of model is recalled.

Definition 11 ([12]). Given an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$ is *satisfied* by I if and only if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a *model* of a MALP $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.

The following definition recalls the generalization of the immediate consequences operator, given by van Emden and Kowalski in [9], to the multi-adjoint logic programming framework.

Definition 12 ([29]). The *immediate consequences operator* $T_{\mathbb{P}}$ maps interpretations to interpretations and it is defined in such a way that, given an interpretation I and a ground atom A,

$$T_{\mathbb{P}}(I)(A) = \sup\{S_I(\vartheta \&_i \mathcal{B}\theta) \mid \langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}, \text{ and } \theta = \text{mgu}\{A, C\}\}$$

where the mapping $S_I \colon \mathfrak{F} \to L$ is defined, for every $\phi \in \mathfrak{F}$, as

$$S_I(\phi) = \sup\{\hat{I}(\phi\xi) \mid \xi \text{ is a ground substitution}\}$$

Next, we will highlight some fundamental properties that this operator satisfies.

Theorem 1 ([24]). *The operator $T_{\mathbb{P}}$ is monotonic.*

Theorem 2 ([24]). *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a model of a MALP $\mathbb{P}$ if and only if $T_{\mathbb{P}}(I) \sqsubseteq I$.*

This operator has a least fixed point as the following result shows.

Proposition 1 ([38]). *The operator $T_{\mathbb{P}} \colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ has a least fixed point $lfp(T_{\mathbb{P}})$ satisfying that*

$$lfp(T_{\mathbb{P}}) = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid T_{\mathbb{P}}(I) \sqsubseteq I\}$$

Taking into account Theorem 2, the least model $M_{\mathbb{P}}$ of the program can be related to the least fixed point of the $T_{\mathbb{P}}$, being possible to obtain it by iterating an ordinal number of times the consequences operator, starting from the least interpretation $\triangle$.

Theorem 3 ([19]). *There exists an ordinal α such that $M_{\mathbb{P}} = lfp(T_{\mathbb{P}}) = T_{\mathbb{P}}^{\alpha}(\triangle)$.*

3. Immediate consequences operators based on general mappings

This section will introduce the semantics associated with a general monotonic immediate consequences operator, which will be denoted as $T_{\mathbb{P}}^{\diamond}$. Notice that we are using the superscript $\diamond$ in order to distinguish this operator from the definition of the operator in the usual multi-adjoint framework, in which it is defined by the supremum operator. Recall that a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$, a multi-adjoint lattice $(L, \preceq, \dot{\&}_1, \ldots, \dot{\&}_m)$, and a MALP $\mathbb{P}$ have been fixed.

In a propositional logic framework, a new definition of immediate consequences operator based on generalized quantifiers was introduced in [26] with the aim of weakening the existential feature of the supremum, to minimize the impact of the possible presence of noise in the data that models the logic rules. This new definition requires of a new notion of model that can capture the essence of the new approach. For this, we presented some examples of generalized quantifiers, for which a declarative definition of model was given. These definitions were designed to solve an specific problem that could be represented by a quantifier. For example, the declarative definition of μ^1-model was given as "I is a μ^1-model if, for each propositional symbol $A \in \Pi$, all the rules with head A are satisfied by I, except possibly one of them", which clearly can be understood as the quantifier Q_{μ^1}, interpreted as "all elements except the greatest one", where the fuzzy measure $\mu^1 \colon \mathcal{P}(\mathbb{P}) \to [0,1]$ is defined as

$$\mu^1(D) = \begin{cases} 0 & \text{if} \quad |D| \leq 1 \\ 1 & \text{otherwise} \end{cases}$$

However, it is not possible to give a general declarative definition, but only particular for each quantifier. That is, we can consider more flexible model representations. Indeed, we can take into account linguistic descriptions given by a proposition p_A depending on an atom A, which can be modelled by an operator $\diamond$ (such as a generalized quantifier or an OWA operator) and such that, an interpretation I satisfies that proposition p_A, for all atom A, if and only if I is a post-fixed point of the immediate consequences operator associated with this operator $\diamond$.

For example, the generalized quantifiers framework introduced in [26] presented linguistic descriptions modeled by generalized quantifiers Q_{μ_A}, such as the example recalled above.

Hence, we introduce the following general definition of $\diamond$-model.

Definition 13. We say that an interpretation I is a $\diamond$-model if and only if $T_{\mathbb{P}}^{\diamond}(I) \sqsubseteq I$. In that case, we say that I satisfies the linguistic description given by the proposition p_A associated with the operator $\diamond$.

Finally, we trivially extend the usual theorem attributed to Kleene, which relates the minimal model $M_{\mathbb{P}}$ with the computation of the least fixed point of the immediate consequences operator.

Theorem 4. *For some ordinal α it is satisfied that $M_{\mathbb{P}} = lfp\,(T_{\mathbb{P}}^{\diamond}) = (T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle)$. In addition, if $T_{\mathbb{P}}^{\diamond}$ is continuous, then $lfp\,(T_{\mathbb{P}}^{\diamond}) = (T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)$, where ω is the smallest limit ordinal.*

Proof. The proof is analogous to the one given in [19]. □

3.1. A particular example of $T_{\mathbb{P}}^{\diamond}$ through OWA operators

Now we are interested in recalling a particular case of $T_{\mathbb{P}}^{\diamond}$ based on ordered weighted averaging aggregation operator [16, 18, 41, 42]. This generalization was analyzed in [27]. In this subsection, we will consider a finite MALP $\mathbb{P}$.

Definition 14 ([42]). Consider a weights vector $W = (w_1, \ldots, w_n)$, satisfying that $w_1 + \ldots + w_n = 1$ and $0 \leq w_i \leq 1$, for all $i \in \{1, \ldots, n\}$. An n-dimensional *ordered weighted averaging aggregation operator*, *OWA operator* in short, is a mapping $@_W \colon [0,1]^n \longrightarrow [0,1]$, associated with W, such that

$$@_W\,(a_1, \ldots, a_n) = \sum_{i=1}^{n} w_i a_{\pi(i)} \tag{3}$$

where π is a permutation of $\{a_1, \ldots, a_n\}$, such that $a_{\pi(n)} \leq \ldots \leq a_{\pi(1)}$.

Now, we recall the following result, which shows that the set of OWA operators is bounded by the maximum OWA (upper bound) and the minimum OWA (lower bound).

Proposition 2 ([42]). *For all OWA operator $@$, it is verified that*

$$@_*\,(a_1, \ldots, a_n) \leq @\,(a_1, \ldots, a_n) \leq @^*\,(a_1, \ldots, a_n)$$

for all $(a_1, \ldots, a_n) \in [0,1]^n$.

Next, we extend the definition of $T_{\mathbb{P}}$ using OWA operators given in [27] to the first order logic setting of this paper.

Definition 15. Let $W \in [0,1]^n$ be a weights vector, satisfying that $w_1 + \ldots + w_n = 1$, and A be an arbitrary ground atom. In addition, consider $\mathbb{P}_A \subseteq \mathbb{P}$, with $|\mathbb{P}_A| = n$, the set composed of all the rules $\langle H_i \leftarrow \mathcal{B}_i; \vartheta_i \rangle$ of $\mathbb{P}$ such that there exists $\theta_i = \text{mgu}\{A, H_i\}$, for all $i \in \{1, \ldots, n\}$. The *OWA immediate consequences operator* $T_{\mathbb{P}}^W \colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$, *on first-order logic* is defined, for A and for all $I \in \mathcal{I}_{\mathfrak{L}}$ as

$$T_{\mathbb{P}}^W(I)(A) \quad = \quad @_W \left(S_I(\vartheta_{\pi(1)} \&_{\pi(1)} \mathcal{B}_{\pi(1)} \theta_{\pi(1)}), \ldots, S_I(\vartheta_{\pi(n)} \&_{\pi(n)} \mathcal{B}_{\pi(n)} \theta_{\pi(n)}) \right) \quad (4)$$

where π is a permutation of $\{1, \ldots, n\}$ satisfying that

$$S_I(\vartheta_{\pi(n)} \&_{\pi(n)} \mathcal{B}_{\pi(n)} \theta_{\pi(n)}) \leq \ldots \leq S_I(\vartheta_{\pi(1)} \&_{\pi(1)} \mathcal{B}_{\pi(1)} \theta_{\pi(1)})$$

Now, the general definition of the $T_{\mathbb{P}}$ based on a family of weights vectors is recalled, where the most appropriate weights vector can be chosen for each ground atom.

Definition 16. Given a family of weights vectors[1]

$$\mathcal{W} = \left\{ W_A \in [0,1]^{n_A} \mid \sum_{i=1}^{n_A} W_A[i] \leq 1, A \text{ is a ground atom}, \mathbb{P}_A \neq \varnothing, n_A = |\mathbb{P}_A| \right\}$$

where $n_A \in \mathbb{N}^*$. We define the $\mathcal{W}$*-OWA immediate consequences operator obtained from the family* $\mathcal{W}$ as the mapping $T_{\mathbb{P}}^{\mathcal{W}} \colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$ defined as

$$T_{\mathbb{P}}^{\mathcal{W}}(I)(A) = \begin{cases} T_{\mathbb{P}}^{W_A}(I)(A) & \text{if} \quad \mathbb{P}_A \neq \varnothing \\ 0 & \text{if} \quad \mathbb{P}_A = \varnothing \end{cases} \quad (5)$$

for all ground atom A and $I \in \mathcal{I}_{\mathfrak{L}}$.

From now on, a family of weights vectors $\mathcal{W}$, according to Definition 16, will be considered. The following result proves the monotonicity of $T_{\mathbb{P}}{}^{\mathcal{W}}$.

[1] $W[i]$ denotes the i-th component of the vector W. Moreover, notice that the definition of OWA can trivially be adapted to consider weights vectors verifying $\sum_{i=1}^{n_A} W_A[i] \leq 1$.

Proposition 3. *The introduced immediate consequences operator $T_{\mathbb{P}}^{\mathcal{W}}$ is increasing in $\mathcal{I}_{\mathfrak{L}}$.*

PROOF. Consider two interpretations $I_1, I_2 \in \mathcal{I}_{\mathfrak{L}}$, such that $I_1 \sqsubseteq I_2$, and a ground atom A. Let $\{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq n\}$ be the maximum subset of rules of $\mathbb{P}$ such that their heads unify with A. The following chain of inequalities holds.

$$
\begin{aligned}
T_{\mathbb{P}}{}^{\mathcal{W}}(I_1)(A) &= T_{\mathbb{P}}{}^{W_A}(I_1)(A) \\
&= @_W\left(S_{I_1}(\vartheta_{\pi(1)}\&_{\pi(1)}\mathcal{B}_{\pi(1)}\theta_{\pi(1)}), \ldots, S_{I_1}(\vartheta_{\pi(n)}\&_{\pi(n)}\mathcal{B}_{\pi(n)}\theta_{\pi(n)})\right) \\
&= \dot{@}_W\Big(\sup\{\vartheta_{\pi(1)}\dot{\&}_{\pi(1)}\hat{I}_1(\mathcal{B}_{\pi(1)}\theta_{\pi(1)}\xi_1) \mid \xi_1 \in \mathcal{S}_g\}, \ldots, \\
&\qquad \sup\{\vartheta_{\pi(n)}\dot{\&}_{\pi(n)}\hat{I}_1(\mathcal{B}_{\pi(n)}\theta_{\pi(n)}\xi_n) \mid \xi_n \in \mathcal{S}_g\}\Big) \\
&\overset{(*)}{\leq} \dot{@}_W\Big(\sup\{\vartheta_{\pi(1)}\dot{\&}_{\pi(1)}\hat{I}_2(\mathcal{B}_{\pi(1)}\theta_{\pi(1)}\xi_1) \mid \xi_1 \in \mathcal{S}_g\}, \ldots, \\
&\qquad \sup\{\vartheta_{\pi(n)}\dot{\&}_{\pi(n)}\hat{I}_2(\mathcal{B}_{\pi(n)}\theta_{\pi(n)}\xi_n) \mid \xi_n \in \mathcal{S}_g\}\Big) \\
&= @_W\left(S_{I_2}(\vartheta_{\pi(1)}\&_{\pi(1)}\mathcal{B}_{\pi(1)}\theta_{\pi(1)}), \ldots, S_{I_2}(\vartheta_{\pi(n)}\&_{\pi(n)}\mathcal{B}_{\pi(n)}\theta_{\pi(n)})\right) \\
&= T_{\mathbb{P}}{}^{W_A}(I_2)(A) \\
&= T_{\mathbb{P}}{}^{\mathcal{W}}(I_2)(A)
\end{aligned}
$$

where (*) follows from the monotonicity of the operators $\dot{\&}$ and $\dot{@}_W$, according to Proposition 2. □

The notion of model must be adapted to this new more versatile framework.

Definition 17. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a *$\mathcal{W}$-model* if, for each ground atom A, I satisfies the linguistic description given by a proposition p_A associated with the corresponding weights vector W_A, that is, $T_{\mathbb{P}}{}^{\mathcal{W}}(I) \sqsubseteq I$.

The least $\mathcal{W}$-model of $\mathbb{P}$, $M_{\mathbb{P}} = \inf\{I \in \mathcal{I}_{\mathfrak{L}} \mid I \text{ is a } \mathcal{W}\text{-model of } \mathbb{P}\}$, can be obtained by iterating $T_{\mathbb{P}}{}^{\mathcal{W}}$ an ordinal number of times, starting from the least interpretation.

Theorem 5. *There exists an ordinal α such that $M_{\mathbb{P}} = \left(T_{\mathbb{P}}^{\mathcal{W}}\right)^{\alpha}(\Delta)$.*

PROOF. The proof is similar to the one given in [27]. □

Finally, we present an example on how the least $\mathcal{W}$-model is calculated.

Example 3. Taking into account Example 2 and Theorem 5, we will see how the least $\mathcal{W}$-model of $\mathbb{P}$ is obtained. For this, we will consider the following family of weights vectors

$$\mathcal{W} \;=\; \{W_A = W_{av} \mid A \text{ is a ground atom}\}$$

where $W_{av} = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$, with $n = |\{\langle H \leftarrow \mathcal{B}; \vartheta\rangle \in \mathbb{P} \mid \text{there exists } \theta = \text{mgu}\{H, A\}\}|$.

$$\begin{array}{rcl}
T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(p(a,b)) & \overset{(1)}{=} & \frac{1}{2} S_{\triangle}\left(0.8 \dot{\&}_{\mathrm{G}} \mathcal{B}_1\theta_1\right) + \frac{1}{2} S_{\triangle}\left(0.8 \dot{\&}_{\mathrm{G}} \mathcal{B}_1\theta_2\right) \\
& \overset{(2)}{=} & \frac{1}{2} \sup\left\{0.8 \dot{\&}_{\mathrm{G}} \widehat{\triangle}(\mathcal{B}_1\theta_1\xi_1^1), 0.8 \dot{\&}_{\mathrm{G}} \widehat{\triangle}(\mathcal{B}_1\theta_1\xi_2^1)\right\} \\
& & + \frac{1}{2} \sup\left\{0.9 \dot{\&}_{\mathrm{P}} \widehat{\triangle}(\mathcal{B}_2\theta_2\xi_1^2), 0.9 \dot{\&}_{\mathrm{P}} \widehat{\triangle}(\mathcal{B}_2\theta_2\xi_2^2)\right\} \\
& = & \frac{1}{2} \sup\left\{0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)}\left(\triangle\left(q(a)\right) \dot{\&}_{\mathrm{G}} \triangle\left(r(b)\right), \triangle\left(s(a,b)\right)\right),\right. \\
& & \qquad \left. 0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)}\left(\triangle\left(q(b)\right) \dot{\&}_{\mathrm{G}} \triangle\left(r(b)\right), \triangle\left(s(b,b)\right)\right)\right\} \\
& & + \frac{1}{2} \sup\left\{0.9 \dot{\&}_{\mathrm{P}}\left(\triangle\left(t(a)\right) \dot{\wedge}_{\mathrm{L}} \triangle\left(r(a)\right)\right),\right. \\
& & \qquad \left. 0.9 \dot{\&}_{\mathrm{P}}\left(\triangle\left(t(a)\right) \dot{\wedge}_{\mathrm{L}} \triangle\left(r(b)\right)\right)\right\} \\
& = & 0 \\
T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(q(a)) & = & 0.2 \dot{\&}_{\mathrm{P}} \widehat{\triangle}(t(a)) \\
& = & 0 \\
T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(q(b)) & = & 0.2 \dot{\&}_{\mathrm{P}} \widehat{\triangle}(t(b)) \\
& = & 0
\end{array}$$

where (1) follows from Definition 15, with the substitutions $\theta_1 = \{X/X', Y/b\}$ and $\theta_2 = \{X/a, Y/Y'\}$. In (2) we have considered the ground substitutions $\xi_1^1 = \{X/a\}$, $\xi_2^1 = \{X/b\}$, $\xi_1^2 = \{Y/a\}$ and $\xi_2^2 = \{Y/b\}$. The second iteration

of $T_{\mathbb{P}}{}^{\mathcal{W}}$ is computed as follows.

$$\begin{aligned}
\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^2(\triangle)(p(a,b)) &= \frac{1}{2}S_{T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)}\left(0.8\dot{\&}_{\mathrm{G}}\mathcal{B}_1\theta_1\right)+\frac{1}{2}S_{T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)}\left(0.8\dot{\&}_{\mathrm{G}}\mathcal{B}_1\theta_2\right)\\
&= \frac{1}{2}\sup\left\{0.8\dot{\&}_{\mathrm{G}}\widehat{T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)}(\mathcal{B}_1\theta_1\xi_1^1),0.8\dot{\&}_{\mathrm{G}}\widehat{T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)}(\mathcal{B}_1\theta_1\xi_2^1)\right\}\\
&\quad+\frac{1}{2}\sup\left\{0.9\dot{\&}_{\mathrm{P}}\widehat{T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)}(\mathcal{B}_2\theta_2\xi_1^2),0.9\dot{\&}_{\mathrm{P}}\widehat{T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)}(\mathcal{B}_2\theta_2\xi_2^2)\right\}\\
&= \frac{1}{2}\sup\Big\{0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\Big(T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(q(a))\dot{\&}_{\mathrm{G}}T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(r(b)),\\
&\qquad T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(s(a,b))\Big),\\
&\qquad 0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\Big(T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(q(b))\dot{\&}_{\mathrm{G}}T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(r(b)),\\
&\qquad T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(s(b,b))\Big)\Big\}\\
&\quad+\frac{1}{2}\sup\Big\{0.9\dot{\&}_{\mathrm{P}}\left(T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(t(a))\dot{\wedge}_{\mathrm{L}}T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(r(a))\right),\\
&\qquad 0.9\dot{\&}_{\mathrm{P}}\left(T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(t(a))\dot{\wedge}_{\mathrm{L}}T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)(r(b))\right)\Big\}\\
&= \frac{1}{2}\sup\left\{0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\left(0\dot{\&}_{\mathrm{G}}0.6,0.8\right),0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\left(0\dot{\&}_{\mathrm{G}}0.6,0\right)\right\}\\
&\quad+\frac{1}{2}\sup\left\{0.9\dot{\&}_{\mathrm{P}}\left(0.6\dot{\wedge}_{\mathrm{L}}0\right),0.9\dot{\&}_{\mathrm{P}}\left(0.6\dot{\wedge}_{\mathrm{L}}0.6\right)\right\}\\
&= 0.225\\
\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^2(\triangle)(q(a)) &= 0.2\dot{\&}_{\mathrm{P}}\widehat{T_{\mathbb{P}}(\triangle)}(t(a))\\
&= 0.12\\
\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^2(\triangle)(q(b)) &= 0.2\dot{\&}_{\mathrm{P}}\widehat{T_{\mathbb{P}}(\triangle)}(t(b))\\
&= 0
\end{aligned}$$

The next table shows the first four iterations of the $T_{\mathbb{P}}{}^{\mathcal{W}}$ for all ground atoms.As the least fixed point of $T_{\mathbb{P}}{}^{\mathcal{W}}$ corresponds to the third iteration, by Theorem 4, the least $\mathcal{W}$-model of $\mathbb{P}$ is exactly $\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^3(\triangle)$.

	$\triangle$	$T_{\mathbb{P}}{}^{\mathcal{W}}(\triangle)$	$\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^2(\triangle)$	$\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^3(\triangle)$	$\left(T_{\mathbb{P}}{}^{\mathcal{W}}\right)^4(\triangle)$
$t(a)$	0	0.6	0.6	0.6	0.6
$s(a,b)$	0	0.8	0.8	0.8	0.8
$r(b)$	0	0.6	0.6	0.6	0.6
$t(b)$	0	0	0	0	0
$s(b,b)$	0	0	0	0	0
$r(a)$	0	0	0	0	0
$q(a)$	0	0	0.12	0.12	0.12
$q(b)$	0	0	0	0	0
$p(a,b)$	0	0	0.225	0.263	0.263
$p(a,a)$	0	0	0.133	0.133	0.133
$p(b,b)$	0	0	0	0	0

In addition, from Expression (2), we can compute the least fixed point of ${T_{\mathbb{P}}}^{\mathcal{W}}$ associated with an arbitrary atom A. For example, if we want to compute the least fixed point associated with $p(a, Y)$, then

$$\begin{aligned}
\widehat{T_{\mathbb{P}}^{\mathcal{W}}(\triangle)}(p(a,Y)) &= \inf\{{T_{\mathbb{P}}}^{\mathcal{W}}(\triangle)(p(a,a)), {T_{\mathbb{P}}}^{\mathcal{W}}(\triangle)(p(a,b))\} \\
&= 0 \\
\left(\widehat{T_{\mathbb{P}}^{\mathcal{W}}(\triangle)}\right)^2(p(a,Y)) &= \inf\{\left({T_{\mathbb{P}}}^{\mathcal{W}}\right)^2(\triangle)(p(a,a)), \left({T_{\mathbb{P}}}^{\mathcal{W}}\right)^2(\triangle)(p(a,b))\} \\
&= 0.133 \\
\left(\widehat{T_{\mathbb{P}}^{\mathcal{W}}(\triangle)}\right)^3(p(a,Y)) &= \inf\{\left({T_{\mathbb{P}}}^{\mathcal{W}}\right)^3(\triangle)(p(a,a)), \left({T_{\mathbb{P}}}^{\mathcal{W}}\right)^3(\triangle)(p(a,b))\} \\
&= 0.133
\end{aligned}$$

□

4. A procedural semantics for multi-adjoint logic programming

The primary objective of this section is to introduce a novel algorithm for obtaining answers to system queries, building upon the approach introduced in [12]. Recall that a *goal* is a body submitted as a query to the system. In addition, a multi-adjoint a multi-adjoint lattice $(L, \preceq, \dot{\&}_1, \ldots, \dot{\&}_m)$, a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$ and a MALP $\mathbb{P}$ have been fixed. The usual notions admissible rules will be considered, whose formal definitions are recalled next [12, 24].

Admissible rules are the basis of the procedural semantics, allowing us to structure the algorithm for query answering by establishing the necessary steps to derive conclusions in a valid and consistent way.

Definition 18 ([12]). Let $\mathcal{Q}$ be a goal and σ be a substitution. The pair $\langle \mathcal{Q}; \sigma \rangle$ is called a state, and the set of states is denoted by ε. An *admissible computation* is formalized as a state transition system, whose transition relation $\rightsquigarrow_{\mathrm{AS}} \subseteq (\varepsilon \times \varepsilon)$ is the smallest relation satisfying the following *admissible rules* (where we consider that A is the selected atom in $\mathcal{Q}$):

- R1. $\langle \mathcal{Q}[A]; \sigma \rangle \rightsquigarrow_{\mathrm{AS}} \langle (\mathcal{Q}[A/\vartheta \&_i \mathcal{B}])\theta; \sigma\theta \rangle$ if $\theta = \mathrm{mgu}\{A, H\}$, $\langle H \leftarrow_i \mathcal{B}; \vartheta \rangle \in \mathbb{P}$.
- R2. $\langle \mathcal{Q}[A]; \sigma \rangle \rightsquigarrow_{\mathrm{AS}} \langle (\mathcal{Q}[A/\bot]); \sigma \rangle$ if there is no rule in $\mathbb{P}$ whose head unifies with A.

Formulas involved in admissible computation steps are renamed apart before being used. The symbol $\rightsquigarrow^*_{\text{AS}}$ denote the reflexive and transitive closure of $\rightsquigarrow_{\text{AS}}$.

Definition 19 ([12]). Let $\mathcal{Q}$ be a goal. An *admissible derivation* is a sequence $\langle \mathcal{Q}; id\rangle \rightsquigarrow^*_{\text{AS}} \langle \mathcal{Q}'; \theta\rangle$. When $\mathcal{Q}'$ is a formula not containing atoms (i.e., it contains only truth values), the pair $\langle \mathcal{Q}'; \sigma\rangle$, where $\sigma = \theta[\mathcal{V}ar(\mathcal{Q})]$, is called an *admissible computed answer* (a.c.a.) for that derivation; where $\theta[\mathcal{V}ar(\mathcal{Q})]$ is the substitution obtained from θ restricting its domain to the set $\mathcal{V}ar(\mathcal{Q})$ of variables occurring in $\mathcal{Q}$.

After successively applying the admissible rules, the goal becomes a formula without atoms and can thus be directly interpreted in the lattice L.

Definition 20 ([12]). Let $\mathcal{Q}$ be a goal and σ a substitution. An *interpretive computation* is a state transition system, whose transition relation $\rightsquigarrow_{\text{IS}} \subseteq (\varepsilon \times \varepsilon)$ is the smallest one satisfying: $\langle \mathcal{Q}[@(r_1, \ldots, r_n)]; \sigma\rangle \rightsquigarrow_{\text{IS}} \langle \mathcal{Q}[@(r_1, \ldots, r_n)/\vartheta]; \sigma\rangle$, where ϑ is the truth value obtained after evaluating $@(r_1, \ldots, r_n)$ in the lattice $\langle L, \preceq\rangle$ associated with $\mathbb{P}$.

Finally, the definition of fuzzy computed answer is recalled.

Definition 21 ([12]). Let $\langle \mathcal{Q}; \sigma\rangle$ be an a.c.a., that is, $\mathcal{Q}$ is a goal not containing atoms. An *interpretive derivation* is a sequence $\langle \mathcal{Q}; \sigma\rangle \rightsquigarrow_{\text{IS}} \langle \mathcal{Q}'; \sigma\rangle$. When $\mathcal{Q}' = r \in L$, the state $\langle r; \sigma\rangle$ is called a *fuzzy computed answer* (f.c.a.) for that derivation.

We denote by $\rightsquigarrow^*_{\text{IS}}$ the reflexive and transitive closure of $\rightsquigarrow_{\text{IS}}$.

Consequently, the procedural semantics arise from applying admissible steps succesively until a formula without atoms, $@(r_1, \ldots, r_n)$, is obtained. The corresponding computed answer is the interpretation of this formula as an element of L, i.e., $\dot{@}\,(r_1, \ldots, r_m)$.

Now, we will focus on the soundness of the procedure. As we are working with a general definition of model, the definition of correct answer must be adapted conveniently

Definition 22. A pair $(\lambda; \theta)$ where $\lambda \in L$ and θ is a substitution, is a *fuzzy $\diamond$-correct answer*, $\diamond$-correct answer in short, for a program $\mathbb{P}$ and a goal $\mathcal{Q}$ if for any $\diamond$-model I of $\mathbb{P}$ we have

$$\lambda \preceq \inf\{I(\mathcal{Q}\theta\xi) \mid \xi \text{ is a ground substitution}\} = \hat{I}(\mathcal{Q}\theta)$$

Note that, for instance, the least element of the lattice $\bot$ is always a fuzzy $\diamond$-correct answer, but it does not give any useful knowledge. This also happens with other (low) values. The following definition introduces the most interesting particular case of $\diamond$-correct answer.

Definition 23. Let θ be a substitution. We define the *θ-maximum $\diamond$-correct answer* for a program $\mathbb{P}$ and a goal $\mathcal{Q}$ as the pair $(\lambda_{\mathrm{M},\theta}; \theta)$, where

$$\lambda_{\mathrm{M},\theta} = \sup \{\lambda \in L \mid (\lambda; \theta) \text{ is a } \diamond\text{-correct answer for } \mathbb{P} \text{ and } \mathcal{Q}\}$$

From the definition of supremum, it is clear that this answer is unique for each goal $\mathcal{Q}$ and substitution θ. Moreover, $\lambda_{\mathrm{M},\theta}$ is, indeed, a $\diamond$-correct answer, as the following result proves.

Lemma 2. *Let $\mathcal{Q}$ be a goal and θ a substitution. Then, $(\lambda_{M,\theta}; \theta)$ is a $\diamond$-correct answer for $\mathbb{P}$ and $\mathcal{Q}$.*

Proof. It follows from Definition 22 and the supremum property. □

Finally, the following theorem states that, for a given substitution θ and goal $\mathcal{Q}$, the θ-maximum $\diamond$-correct answer can be obtained by iterating $T^{\diamond}_{\mathbb{P}}$ an ordinal number of times.

Theorem 6. *Let $\mathcal{Q}$ be a goal, θ a substitution in $\mathcal{Q}$. Then, there exists an ordinal α such that $\left(\widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta); \theta\right)$ is the θ-maximum $\diamond$-correct answer for $\mathcal{Q}$ and $\mathbb{P}$.*

Proof. On the one hand, from Theorem 4, there exists an ordinal α such that $(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)$ is the least $\diamond$-model of $\mathbb{P}$ and, in particular, it is a $\diamond$-model of $\mathbb{P}$. From Definition 22, inequality $\lambda \preceq \widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta)$ holds for every $\diamond$-correct answer $(\lambda; \theta)$. Now, from the supremum property, it is clear that $\lambda_{\mathrm{M},\theta} \preceq \widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta)$.

On the other hand, as $(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)$ is the least $\diamond$-model of $\mathbb{P}$, then $\widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta) \preceq \hat{I}(\mathcal{Q}\theta)$, for all $\diamond$-model $I \in \mathcal{I}_{\mathfrak{L}}$. Hence, $\left(\widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta); \theta\right)$ is a $\diamond$-correct answer for $\mathcal{Q}$. From Definition 23, it is satisfied that $\widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta) \preceq \lambda_{\mathrm{M},\theta}$. Thus, we obtain that $\widehat{(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle)}(\mathcal{Q}\theta) = \lambda_{\mathrm{M},\theta}$. □

Note that, by the Kleene's theorem (Theorem 4), an ordinal α always exists, such that $(T^{\diamond}_{\mathbb{P}})^{\alpha}(\triangle) = \mathrm{lfp}(T^{\diamond}_{\mathbb{P}})$. One interesting remark is that sometimes the most interesting answer, which is the θ-maximum $\diamond$-correct answer, for a given θ and goal $\mathcal{Q}$, cannot be obtained through computed answers [31]. This situation motivates the following section.

4.1. A new definition of reductants adapted to this framework

In order to solve the problem aforementioned, a new definition of reductant in this new framework will be studied. Recall that we have fixed a multi-adjoint lattice $(L, \preceq, \dot{\&}_1, \ldots, \dot{\&}_m)$, a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$ and a MALP $\mathbb{P}$. First of all, we recall the definition of critical rules, followed by the new definition of reductant.

Definition 24 ([12]). Let $\mathcal{R}_1 = \langle H_1 \leftarrow \mathcal{B}_1, \vartheta_1 \rangle$ and $\mathcal{R}_2 = \langle H_2 \leftarrow \mathcal{B}_2, \vartheta_2 \rangle$ be two rules in $\mathbb{P}$ that are renamed-apart. The rules $\mathcal{R}_1$ and $\mathcal{R}_2$ are said to be *critical* if and only if H_1 and H_2 unify, that is, there exists a substitution $\theta = \mathrm{mgu}\{H_1, H_2\}$. In this case, we say that these critical rules are associated with $H_1\theta$.

A set of rules in $\mathbb{P}$ which are renamed-apart is said to be *a set of critical rules* if and only if the set of their heads unify.

In addition, given a ground atom A, we will define the *maximum set of critical rules in* $\mathbb{P}$ *associated with* A, denoted as C_M^A, as the set of all the rules in $\mathbb{P}$ whose heads unify with A.

Definition 25. Let $C = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i \rangle \mid 1 \leq i \leq r\}$, with $\theta = \mathrm{mgu}\{H_1, \ldots, H_r\}$, be a set of critical rules in $\mathbb{P}$, such that $|\mathcal{S}_g(\mathcal{B}_i\theta)| = n_i \in \mathbb{N}$, for all $i \in \{1, \ldots, r\}$. Then the rule

$$\begin{aligned}\big\langle H_1\theta\theta_1 \leftarrow @(&@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots,\\ &@_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r)); \top\big\rangle\end{aligned}$$

is said to be a *C-critical $\diamond$-reductant* of $\mathbb{P}$, in short C-$\diamond$-reductant or $\diamond$-reductant associated with C, where $\theta_1 \in \mathcal{S}_g(H_1\theta)$, $\xi_1^i, \ldots, \xi_{n_i}^i \in \mathcal{S}_g(\mathcal{B}_i\theta\theta_1)$, for all $i \in \{1, \ldots, r\}$, the connective $\leftarrow$ is any implication with an adjoint conjunctor, $@_{\sup}$ is defined as

$$\dot{@}_{\sup}(b_1, \ldots, b_{n_i}) = \sup\{b_1, \ldots, b_{n_i}\}$$

for any $b_1, \ldots, b_{n_i} \in L$, and the truth function for the symbol @ is a weak aggregator satisfying

$$T_{\mathbb{P}}^{\diamond}(I)(H_1\theta\theta_1) \;=\; \dot{@}\,(w_1, \ldots, w_r) \tag{6}$$

where $w_i = \dot{@}_{\sup}(\vartheta_i \dot{\&}_i b_1^i, \ldots, \vartheta_i \dot{\&}_i b_{n_i}^i)$, for all $i \in \{1, \ldots, r\}$, and I is an interpretation such that

$$I(\mathcal{B}_i\theta\theta_1\xi_1^i) = b_1^i, \ldots, I(\mathcal{B}_i\theta\theta_1\xi_{n_i}^i) = b_{n_i}^i, \text{ for all } i \in \{1, \ldots, r\}$$

that is,

$$S_I\,(\vartheta_i \&_i \mathcal{B}_i\theta\theta_1) = \dot{@}_{\sup}(\vartheta_i \dot{\&}_i b_1^i, \ldots, \vartheta_i \dot{\&}_i b_{n_i}^i) = w_i, \text{ for all } i \in \{1, \ldots, r\}$$

The critical $\diamond$-reductant for an arbitrary ground atom will be defined through the maximum set of critical rules in $\mathbb{P}$ associated with the corresponding ground atom.

Definition 26. Given a ground atom A, we define *the (maximum) critical $\diamond$-reductant for A*, in short $\diamond$-reductant for A, as the C_M^A-critical $\diamond$-reductant.

Consider the set of critical rules in $\mathbb{P}$, $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq r\}$, with $\theta = \text{mgu}\{H_1, \ldots, H_r\}$. Notice that, if $|\mathcal{S}_g(H_1\theta)| = n$, then we have n different $\diamond$-reductants associated with $C_M^{H_1\theta}$, and each one is the critical $\diamond$-reductant for the ground atom $H_1\theta\theta_1$, with $\theta_1 \in \mathcal{S}_g(H_1\theta)$, according to Definition 26. In addition, we could also have several more $\diamond$-reductants for every set of critical rules, conveniently modifying the aggregator @. However, in this case, all of them provide the same information, given by the previous Equality (6). Moreover, as the $\diamond$-reductant for each ground atom A is associated with C_M^A, which is unique, then we can ensure that the $\diamond$-reductant for each ground atom in $\mathbb{P}_r$ is unique. Consequently, we can indeed refer as *the* C_M^A-$\diamond$-reductant for the ground atom A, in Definition 26.

The following example shows that, although the $\diamond$-reductant is a fact, not all the rules have to be facts.

Example 4. Let $\mathbb{P}$ be a MALP given by the rules $\langle H_1 \leftarrow_1 \top, \top\rangle$ and $\langle H_2 \leftarrow_2 B, 0.1\rangle$, where H_1 and H_2 are atoms, B is a ground atom, $\theta = \text{mgu}\{H_1, H_2\}$; and W^* be the weights vector related to the maximum operator, that is, $W^* = (1, 0)$. From Definition 25, a $\mathcal{W}$-reductant for these rules is

$$\langle H_1\theta \leftarrow @\left(@_{\sup}(\top \&_1 \top), @_{\sup}(0.1 \&_2 B)\right)\theta; \top\rangle$$

where the operator $\dot{@}$ satisfies

$$\dot{@}\left(\dot{@}_{\sup}(\top \dot{\&}_1 \top), \dot{@}_{\sup}(0.1 \dot{\&}_2 b)\right) = \dot{@}\left(\top, 0.1 \dot{\&}_2 b\right) = \widehat{T_{\mathbb{P}}^{\mathcal{W}^*}(I)}(H_1\theta) = \top$$

with the interpretation I satisfying that $I(B\theta) = b$. Consequently, the $\diamond$-reductant is $\langle H_1\theta \leftarrow \top; \top\rangle$, but not all the matching rules are facts. □

Now, we present a technical result which shows that the number of ground substitutions is not going to change, when the set of free variables is the same.

Lemma 3. *Let $\mathcal{B}$ be a formula, and θ, θ' be two substitutions. If $\mathcal{V}ar_l(\mathcal{B}\theta) = \mathcal{V}ar_l(\mathcal{B}\theta')$, then $\mathcal{S}_g(\mathcal{B}\theta) = \mathcal{S}_g(\mathcal{B}\theta')$.*

PROOF. It directly follows from the definitions of $\mathcal{V}ar_l$ and $\mathcal{S}_g$ (Expression (1)). □

The importance of the next result lies in the fact that if we add the $\diamond$-reductants to the program $\mathbb{P}$, the set of $\diamond$-models is not going to change, and so the main information in $\mathbb{P}$ is preserved.

Proposition 4. *Any $\diamond$-reductant according to Definition 25 is satisfied by any $\diamond$-model of $\mathbb{P}$.*

PROOF. Consider a $\diamond$-model I of $\mathbb{P}$ and an arbitrary critical $\diamond$-reductant $\langle H_1\theta\theta_1 \leftarrow \mathcal{B}\theta\theta_1; \top\rangle$. From Definition 25, there exists a set $C = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq r\}$ of critical rules in $\mathbb{P}$, with $\theta = \text{mgu}\{H_1, \ldots, H_r\}$ and $\theta_1 \in \mathcal{S}_g(H_1\theta)$, such that $\langle H_1\theta\theta_1 \leftarrow \mathcal{B}\theta\theta_1; \top\rangle$ is one critical $\diamond$-reductant associated with C, where $\mathcal{B}\theta\theta_1$ is given by

$$\bar{@}(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))$$

We must prove that $\top \preceq \hat{I}(H_1\theta\theta_1 \leftarrow \mathcal{B}\theta\theta_1)$ or, equivalently, $\top \dot{\&} \hat{I}(\mathcal{B}\theta\theta_1) \preceq I(H_1\theta\theta_1)$. The following chain of inequalities holds.

$$\begin{aligned}
\top \dot{\&} \hat{I}(\mathcal{B}\theta\theta_1) &= \hat{I}\left(@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, \right. \\
&\qquad \left. @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\right) \\
&= \dot{@}(\dot{@}_{\sup}(\hat{I}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \hat{I}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1)), \ldots, \\
&\qquad \dot{@}_{\sup}(\hat{I}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \hat{I}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))) \\
&= \dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_{n_1}^1)), \ldots, \\
&\qquad \dot{@}_{\sup}(\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r))) \\
&\overset{(1)}{=} \dot{@}\left(S_I(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1), \ldots, S_I(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1)\right) \\
&\overset{(2)}{=} T_{\mathbb{P}}^{\diamond}(I)(H_1\theta\theta_1) \\
&\overset{(3)}{\preceq} I(H_1\theta\theta_1)
\end{aligned}$$

where (1) is satisfied due to the definition of the operator S_I, (2) holds from the definition of $\diamond$-reductant and (3) follows from the fact that I is a $\diamond$-model and, consequently, $T_{\mathbb{P}}^{\diamond}(I) \sqsubseteq I$. □

Given a program $\mathbb{P}$, we will introduce a new program $\mathbb{P}_r$ associated with $\mathbb{P}$, consisting of the critical $\diamond$-reductant for every ground atom associated with $\mathbb{P}$, in such a way that both programs are semantically equivalent.

Definition 27. We define the *program by $\diamond$-reductants for* $\mathbb{P}$ or r-program in short, denoted by $\mathbb{P}_r$, as a MALP consisting of the $\diamond$-reductant for each ground atom A, such that A unifies with the head of a rule in $\mathbb{P}$.

Note that, the program $\mathbb{P}_r$ is obtained from $\mathbb{P}$ computing the corresponding $\diamond$-reductants, preserving the main information. The following result relates the notion of unification between the two programs $\mathbb{P}$ and $\mathbb{P}_r$.

Proposition 5. *Let A be an atom. Then, A matches with the head of a rule in $\mathbb{P}_r$ if and only if A matches with the head of a rule in $\mathbb{P}$.*

Proof. Consider an atom A. On the one hand, if A matches with the head of a rule in $\mathbb{P}_r$, then there exists a ground atom $H_1\theta\theta_1$, head of a rule in $\mathbb{P}_r$, which comes from the set of critical rules in $\mathbb{P}$, $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq r\}$, with $\theta = \text{mgu}\{H_1, \ldots, H_r\}$, $\theta_1 \in \mathcal{S}_g(H_1\theta)$, and there exists a substitution ν such that $A\nu = H_1\theta\theta_1$. Notice that A and $H_1\theta$ are standardized apart, and so θ_1 and ν substitute different variables, which implies that $\theta_1\nu = \nu\theta_1$. As a consequence, we have that

$$A\theta_1\nu = A\nu\theta_1 \overset{(*)}{=} A\nu = H_1\theta\theta_1 \overset{(*)}{=} H_1\theta\theta_1\nu$$

where (*) follows from the fact that $A\nu$ and $H_1\theta\theta_1$ are ground atoms. Hence, A matches with the head of a rule in $\mathbb{P}$.

On the other hand, if A matches with the head of a rule in $\mathbb{P}$, say H_1, then there exists a substitution ν such that $A\nu = H_1\nu$. Considering $\theta_1 \in \mathcal{S}_g(H_1\nu)$, we have a rule in $\mathbb{P}_r$ with head $H_1\nu\theta_1$. Therefore, A matches with the head of a rule in $\mathbb{P}_r$. □

An example of r-program will now be provided.

Example 5. Consider the program $\mathbb{P}$ introduced in Example 2 and the family of weights vectors $\mathcal{W}$ given in Example 3. Following Definition 27, the r-program $\mathbb{P}_r$ for $\mathbb{P}$ has the following rules associated with the maximum sets of critical rules $C_M^{p(a,b)}$, $C_M^{p(a,Y)}$, $C_M^{p(X,b)}$ and $C_M^{q(X)}$.

- For $p(a,b)$, the $C_M^{p(a,b)}$-$\diamond$-reductant is

$$\begin{aligned}\mathcal{R}_1' \;=\; \big\langle p(a,b) \leftarrow @\,\big(@_{\text{sup}}\,\big(&0.8 \&_{\text{G}} @_{(2,1)}(q(a) \&_{\text{G}} r(b), s(a,b)),\\ &0.8 \&_{\text{G}} @_{(2,1)}(q(b) \&_{\text{G}} r(b), s(b,b))\big),\\ &@_{\text{sup}}\,(0.9 \&_{\text{P}} (t(a) \wedge_{\text{Ł}} r(a)), 0.9 \&_{\text{P}} (t(a) \wedge_{\text{Ł}} r(b)))\big)\,;\top\big\rangle\end{aligned}$$

where $\dot{@}\left(\dot{@}_{\sup}(0.8\dot{\&}_{\mathrm{G}}b_1^1, 0.8\dot{\&}_{\mathrm{G}}b_2^1), \dot{@}_{\sup}(0.9\dot{\&}_{\mathrm{P}}b_2^1, 0.9\dot{\&}_{\mathrm{P}}b_2^2)\right)$ is equal to

$$T_{\mathbb{P}}{}^{\mathcal{W}}(I)(p(a,b)) = \frac{1}{2}\dot{@}_{\sup}(0.8\dot{\&}_{\mathrm{G}}b_1^1, 0.8\dot{\&}_{\mathrm{G}}b_2^1) + \frac{1}{2}\dot{@}_{\sup}(0.9\dot{\&}_{\mathrm{P}}b_2^1, 0.9\dot{\&}_{\mathrm{P}}b_2^2)$$

with the interpretation I satisfying that

$$\hat{I}(@_{(2,1)}(q(a)\&_{\mathrm{G}}r(b), s(a,b))) = b_1^1 \qquad \hat{I}(t(a) \wedge_{\mathrm{L}} r(a)) = b_2^1$$
$$\hat{I}(@_{(2,1)}(q(b)\&_{\mathrm{G}}r(b), s(b,b))) = b_2^1 \qquad \hat{I}(t(a) \wedge_{\mathrm{L}} r(b)) = b_2^2$$

- As $|\,\mathcal{S}_g(p(a,Y))| = 2$, then we have two $\diamond$-reductants associated with $C_M^{p(a,Y)}$. The one with head $p(a,b)$ has already been presented. For $p(a,a)$, the $C_M^{p(a,a)}$-$\diamond$-reductant is

$$\mathcal{R}_2' \;=\; \langle p(a,a) \leftarrow @'\left(@_{\sup}\left(0.8\&_{\mathrm{G}}@_{(2,1)}(q(a)\&_{\mathrm{G}}r(a), s(a,b)),\right.\right.$$
$$\left.\left.0.8\&_{\mathrm{G}}@_{(2,1)}(q(b)\&_{\mathrm{G}}r(a), s(b,b))\right)\right) ; \top\rangle$$

where

$$\dot{@}'\left(\dot{@}_{\sup}(0.8\dot{\&}_{\mathrm{G}}b_1^1, 0.8\dot{\&}_{\mathrm{G}}b_2^1)\right) = T_{\mathbb{P}}{}^{\mathcal{W}}(I)(p(a,a)) = \dot{@}_{\sup}(0.8\dot{\&}_{\mathrm{G}}b_1^1, 0.8\dot{\&}_{\mathrm{G}}b_2^1)$$

with the interpretation I satisfying that

$$\hat{I}(@_{(2,1)}(q(a)\&_{\mathrm{G}}r(a), s(a,b))) = b_1^1 \qquad \hat{I}(@_{(2,1)}(q(b)\&_{\mathrm{G}}r(a), s(b,b))) = b_2^1$$

- As $|\,\mathcal{S}_g(p(X,b))| = 2$, then we have two $\diamond$-reductants associated with $C_M^{p(X,b)}$. As we have previously commented, since the $\diamond$-reductant with head $p(a,b)$ has already been considered, we compute now $\diamond$-reductant for $p(b,b)$. The $C_M^{p(b,b)}$-$\diamond$-reductant is

$$\mathcal{R}_3' \;=\; \langle p(b,b) \leftarrow @''\left(@_{\sup}\left(0.9\&_{\mathrm{P}}(t(b) \wedge_{\mathrm{L}} r(a)), 0.9\&_{\mathrm{P}}(t(b) \wedge_{\mathrm{L}} r(b))\right)\right) ; \top\rangle$$

where

$$\dot{@}''\left(\dot{@}_{\sup}(0.9\dot{\&}_{\mathrm{P}}b_1^2, 0.9\dot{\&}_{\mathrm{P}}b_2^2)\right) = T_{\mathbb{P}}{}^{\mathcal{W}}(I)(p(b,b)) = \dot{@}_{\sup}(0.9\dot{\&}_{\mathrm{P}}b_1^2, 0.9\dot{\&}_{\mathrm{P}}b_2^2)$$

with the interpretation I satisfying that

$$\hat{I}(t(a) \wedge_{\mathrm{L}} r(a)) = b_2^1 \qquad \hat{I}(t(a) \wedge_{\mathrm{L}} r(b)) = b_2^2$$

- As $|\,\mathcal{S}_g(q(X))| = 2$, we have two $\diamond$-reductants associated with $C_M^{q(X)}$. On the one hand, for $q(a)$, the $C_M^{q(a)}$-$\diamond$-reductant is

$$\mathcal{R}_4' \;=\; \langle q(a) \leftarrow @'''\,(0.2 \&_{\mathrm{P}} t(a))\,;\top\rangle$$

where $\dot{@}'''(0.2\dot{\&}_{\mathrm{P}} b_1^3) = T_{\mathbb{P}}^{\diamond}(I)(q(a)) = 0.2\dot{\&}_{\mathrm{P}} b_1^3$, with the interpretation I satisfying that $I(t(a)) = b_1^3$. On the other hand, for $q(b)$, the $C_M^{q(b)}$-$\diamond$-reductant is

$$\mathcal{R}_5' \;=\; \langle q(b) \leftarrow @''''\,(0.2 \&_{\mathrm{P}} t(b))\,;\top\rangle$$

where $\dot{@}''''(0.2\dot{\&}_{\mathrm{P}} b_2^3) = T_{\mathbb{P}}^{\diamond}(I)(q(b)) = 0.2\dot{\&}_{\mathrm{P}} b_2^3$, with the interpretation I satisfying that $I(t(b)) = b_2^3$.

This program $\mathbb{P}_r$ is completed with the same facts as those of $\mathbb{P}$, since, for an atom A head of a fact in $\mathbb{P}$, $|\,C_M^A\,| = 1$. Consequently, the r-program $\mathbb{P}_r$ associated with $\mathbb{P}$ is

$$\mathbb{P}_r = \{\mathcal{R}_1', \mathcal{R}_2', \mathcal{R}_3', \mathcal{R}_4', \mathcal{R}_5', \mathcal{R}_4, \mathcal{R}_5, \mathcal{R}_6\}$$

□

The following result establishes that, indeed, both programs $\mathbb{P}$ and $\mathbb{P}_r$ are semantically equivalent.

Proposition 6. *Let $\mathbb{P}$ be a program and $\mathbb{P}_r$ its associated r-program. Then, I is a $\diamond$-model of $\mathbb{P}$ if and only if I is a model of $\mathbb{P}_r$.*

Proof. On the one hand, consider a $\diamond$-model of $\mathbb{P}$ $I \in \mathcal{I}_{\mathfrak{L}}$. From Proposition 4, it is satisfied that I is a model of $\mathbb{P}_r$.

On the other hand, consider a model of $\mathbb{P}_r$ $I \in \mathcal{I}_{\mathfrak{L}}$ and a ground atom A. We have to prove that $T_{\mathbb{P}}^{\diamond}(I) \sqsubseteq I$.

If A does not match with the head of any rule in $\mathbb{P}$, then the inequality $T_{\mathbb{P}}^{\diamond}(I)(A) \preceq I(A)$ trivially holds. Otherwise, A matches with the head of some rule in $\mathbb{P}$. In this case, there must exist a set $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq r\}$ in $\mathbb{P}$, with $\theta = \mathrm{mgu}\{H_1, \ldots, H_r\}$, such that there exists a substitution $\theta_1 \in \mathcal{S}_g(H_1\theta)$ verifying that $A = H_1\theta\theta_1$. In this situation, the following critical $\diamond$-reductant

$$\big\langle H_1\theta\theta_1 \leftarrow @(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r)), \top\big\rangle$$

can be considered in $\mathbb{P}_r$, with $\xi_1^i, \ldots, \xi_{n_i}^i \in \mathcal{S}_g(\mathcal{B}_i\theta\theta_1)$, for all $i \in \{1, \ldots, r\}$. The following chain of inequalities is obtained.

$$\begin{aligned}
T_{\mathbb{P}}^{\diamond}(I)(A) &= T_{\mathbb{P}}^{\diamond}(I)(H_1\theta\theta_1) \\
&\overset{(1)}{=} \dot{@}\Big(\dot{@}_{\sup}(\vartheta_1\dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \vartheta_1\dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^{n_1})), \ldots, \\
&\qquad \dot{@}_{\sup}(\vartheta_1\dot{\&}_1 I(\mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \vartheta_r\dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\Big) \\
&\overset{(2)}{=} \sup\Big\{\dot{@}\Big(\dot{@}_{\sup}(\vartheta_1\dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \vartheta_1\dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^{n_1})), \ldots, \\
&\qquad \dot{@}_{\sup}(\vartheta_1\dot{\&}_1 I(\mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \vartheta_r\dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\Big)\, \nu \mid \nu \in \mathcal{S}_g\Big\} \\
&= T_{\mathbb{P}_r}(I)(A) \\
&\overset{(3)}{\preceq} I(A)
\end{aligned}$$

where (1) follows from the definition of $\diamond$-reductant, (2) holds because the formula

$$\begin{aligned}
&\dot{@}\Big(\dot{@}_{\sup}(\vartheta_1\dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \vartheta_1\dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi_1^{n_1})), \ldots, \\
&\quad \dot{@}_{\sup}(\vartheta_1\dot{\&}_1 I(\mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \vartheta_r\dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\Big)
\end{aligned}$$

there is no variables. Finally, (3) is true because I is a model of $\mathbb{P}_r$. Therefore, we have proved that the inequality

$$T_{\mathbb{P}}^{\diamond}(I)(A) \preceq I(A) \tag{7}$$

holds for all ground atom A and model I of $\mathbb{P}_r$. Now, given a general atom A, applying Expression (7), we obtain that

$$\widehat{T_{\mathbb{P}}^{\diamond}(I)}(A) = \inf\{T_{\mathbb{P}}^{\diamond}(I)(A\xi) \mid \xi \in \mathcal{S}_g(A)\} \preceq \inf\{I(A\xi) \mid \xi \in \mathcal{S}_g(A)\} = \hat{I}(A)$$

We can conclude that I is a $\diamond$-model of $\mathbb{P}$. □

Thus, the program $\mathbb{P}_r$ offers an alternative perspective on the original program $\mathbb{P}$, providing a different approach for computing the $\diamond$-models of $\mathbb{P}$.

The procedural semantics presented in this paper also requires a new definition of computed answer for a query, as we can see with the following example.

Example 6. Consider the program $\mathbb{P}$ introduced in Example 2, the family of weights vectors $\mathcal{W}$ from Example 3 and the interpretation I given by

$$\begin{array}{ccc}
I(r(a)) = 0 & I(r(b)) = 0.6 & I(s(a,b)) = 0.8 \\
I(s(b,b)) = 0 & I(t(a)) = 0.6 & I(t(b)) = 0 \\
I(q(a)) = 0.2 & I(q(b)) = 0 & I(p(a,b)) = 0.3
\end{array}$$

which is a $\mathcal{W}$-model of $\mathbb{P}$. A fuzzy computed answer for $?p(a,b)$ is computed as follows.

$$
\begin{array}{rl}
\langle p(a,b); id\rangle & \overset{\mathcal{R}_1}{\rightsquigarrow}_{\mathrm{AS}} \ \left\langle 0.8 \&_{\mathrm{G}} @_{(2,1)} \left(q(X) \&_{\mathrm{G}} r(b), s(X,b)\right); \{Y/b\}\right\rangle \\
& \overset{\mathcal{R}_3}{\rightsquigarrow}_{\mathrm{AS}} \ \left\langle 0.8 \&_{\mathrm{G}} @_{(2,1)} \left(\left(0.2 \&_{\mathrm{P}} t(X)\right) \&_{\mathrm{G}} r(b), s(X,b)\right); \{Y/b\}\right\rangle \\
& \overset{\mathcal{R}_6}{\rightsquigarrow}_{\mathrm{AS}} \ \left\langle 0.8 \&_{\mathrm{G}} @_{(2,1)} \left(\left(0.2 \dot{\&}_{\mathrm{P}} 0.6\right) \&_{\mathrm{G}} r(b), s(a,b)\right); \{X/a, Y/b\}\right\rangle \\
& \overset{\mathcal{R}_4}{\rightsquigarrow}_{\mathrm{AS}} \ \left\langle 0.8 \&_{\mathrm{G}} @_{(2,1)} \left(\left(0.2 \dot{\&}_{\mathrm{P}} 0.6\right) \dot{\&}_{\mathrm{G}} 0.6, s(a,b)\right); \{X/a, Y/b\}\right\rangle \\
& \overset{\mathcal{R}_5}{\rightsquigarrow}_{\mathrm{AS}} \ \left\langle 0.8 \dot{\&}_{\mathrm{G}} \dot{@}_{(2,1)} \left(\left(0.2 \dot{\&}_{\mathrm{P}} 0.6\right) \dot{\&}_{\mathrm{G}} 0.6, 0.8\right); \{X/a, Y/b\}\right\rangle \\
& \rightsquigarrow_{\mathrm{IS}} \ \langle 0.34; \sigma\rangle
\end{array}
$$

where $\sigma = \{X/a, Y/b\}\,[\mathcal{V}ar(p(a,b))] = \{X/a, Y/b\}\,[\varnothing] = id$. Therefore, $\langle 0.34; id\rangle$ is a computed answer for $?p(a,b)$, but it is not a $\mathcal{W}$-correct answer for $?p(a,b)$ since

$$0.34 \not\leq 0.3 = I(p(a,b))$$

for the previous $\mathcal{W}$-model I. □

Then, it is clear that the definition of computed answer (Definition 21) is not suitable for the flexible framework in which we are working. In order to solve this, the following definition adaptes the notion of computed answer for a query to the approach considered in this paper.

Definition 28. Let $\mathbb{P}_r$ be the r-program of $\mathbb{P}$ and $\mathcal{Q}$ be a goal. We say that $\langle \mathcal{A}; \sigma\rangle$, where $\mathcal{A}$ is a formula not containing atoms and σ is a substitution, is an *admissible $\mathbb{P}_r$-computed answer* if it is an admissible computed answer with respect to $\mathbb{P}_r$ for the same goal $\mathcal{Q}$. Furthermore, we say that $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$, where $\dot{\mathcal{A}} \in L$, is a *fuzzy $\mathbb{P}_r$-computed answer for $\mathcal{Q}$*, if it is a fuzzy computed answer for $\mathcal{Q}$ with respect to $\mathbb{P}_r$.

The natural question that arises is whether a computed answer based on Definition 28 is correct according to Definition 22. Next section will answer this question positively.

5. Soundness of the proposed procedural semantics

The soundness will be based on the notion of fuzzy $\mathbb{P}_r$-computed answer for a given query. Hence, first of all we need to prove the equivalence

between the notions of $\diamond$-correct answer for $\mathbb{P}$ and correct answer for $\mathbb{P}_r$, showing that the program $\mathbb{P}_r$ preserves the $\diamond$-correct answers from $\mathbb{P}$. Recall that a MALP $\mathbb{P}$, a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$ and a multi-adjoint lattice $\mathcal{L} = \left\langle L, \preceq, \dot{\&}_1, \ldots, \&_m \right\rangle$ have been fixed.

Proposition 7. *Consider the r-program $\mathbb{P}_r$ of $\mathbb{P}$ and a goal $\mathcal{Q}$. Then, $(\lambda; \theta)$ is a fuzzy $\diamond$-correct answer for $\mathbb{P}$ and $\mathcal{Q}$ if and only if $(\lambda; \theta)$ is a fuzzy correct answer for $\mathbb{P}_r$ and $\mathcal{Q}$.*

Proof. The proof follows from the following chain of equivalencies. $(\lambda; \theta)$ is a fuzzy $\diamond$-correct answer for $\mathbb{P}$ and $\mathcal{Q}$ if and only if $\lambda \preceq \hat{I}(\mathcal{Q}\theta)$, for all $\diamond$-model I of $\mathbb{P}$, which is equivalent, by Proposition 6, to $\lambda \preceq \hat{I}(\mathcal{Q}\theta)$ for all model I of $\mathbb{P}_r$, that is, $(\lambda; \theta)$ is a fuzzy correct answer for $\mathcal{Q}$ in $\mathbb{P}_r$. □

The following theorem establishes the soundness of the new procedural semantics, allowing us to ensure that the fuzzy $\mathbb{P}_r$-computed answers according to Definition 28 are correct.

Theorem 7. *Let A be an atom and $\mathbb{P}_r$ be the r-program of $\mathbb{P}$. Then, every fuzzy $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ for $?A$ is a fuzzy $\diamond$-correct answer $(\dot{\mathcal{A}}; \sigma)$ for $\mathbb{P}$ and $?A$, where σ is the most general unifier of A and the head of a rule in $\mathbb{P}_r$.*

Proof. Consider an atom A and a fuzzy $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$. We are going to prove the result by induction over the length n of the computation sequence of the admissible $\mathbb{P}_r$-computed answer $\langle \mathcal{A}; \sigma \rangle$, from which the fuzzy $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ for $?A$ is obtained through the interpretive step. First of all, assume that it is obtained in $n = 1$ steps. For this case, the following possibilities arise.

1. The atom A does not match with the head of any rule in $\mathbb{P}_r$. The corresponding sequence for obtaining $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ is

   $$\langle A; id \rangle \rightsquigarrow_{\text{AS}} \langle \bot; id \rangle \rightsquigarrow_{\text{IS}} \langle \bot; id \rangle$$

   which is always a fuzzy $\diamond$-correct answer for $?A$.

2. The atom A matches with the head $H_1\theta\theta_1$ of a rule in $\mathbb{P}_r$. In this case, as $n = 1$, the rule of $\mathbb{P}_r$ must come from a set of critical facts $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \top; \vartheta_i\rangle \in \mathbb{P} \,|\, 1 \leq i \leq r\}$, with $\theta = \text{mgu}\{H_1, \ldots, H_r\}$ and $\theta_1 \in \mathcal{S}_g(H_1\theta)$. Therefore, the rule in $\mathbb{P}_r$ is

$$\langle H_1\theta\theta_1 \leftarrow @(\vartheta_1, \ldots, \vartheta_r); \top\rangle$$

and there exists a substitution ν such that $A\nu = H_1\theta\theta_1$. The fuzzy $\mathbb{P}_r$-computed answer follows from the sequence

$$\langle A; id\rangle \rightsquigarrow_{\text{AS}} \langle @(\vartheta_1, \ldots, \vartheta_r); \nu\rangle \rightsquigarrow_{\text{IS}} \left\langle \dot{@}(\vartheta_1, \ldots, \vartheta_r); \nu \right\rangle$$

where, from the definition of $\diamond$-reductant, it is satisfied that $\dot{@}(\vartheta_1, \ldots, \vartheta_r) = T_{\mathbb{P}}^{\diamond}(I)(H_1\theta\theta_1)$ with I any interpretation. Therefore, we can consider a $\diamond$-model I of $\mathbb{P}$ and then,

$$\dot{@}(\vartheta_1, \ldots, \vartheta_r) = T_{\mathbb{P}}^{\diamond}(I)(H_1\theta\theta_1) \preceq I(H_1\theta\theta_1) = I(A\nu)$$

Consequently, $\left\langle \dot{@}(\vartheta_1, \ldots, \vartheta_r); \nu \right\rangle$ is a fuzzy $\diamond$-correct answer for $?A$.

Suppose that the result is true for all length of computation less than n, with $1 < n$, that is, if $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for a query $?C$, such that the length of computation of the associated admissible $\mathbb{P}_r$-computed answer $\langle \mathcal{A}; \sigma\rangle$ is $k < n$, then $(\dot{\mathcal{A}}; \sigma)$ is a fuzzy $\diamond$-correct answer for $?C$. Now, we will prove it for $k + 1$. For this, consider that the length of computation of the admissible $\mathbb{P}_r$-computed answer $\langle \mathcal{A}; \sigma\rangle$ is $k + 1$. We will see that $(\dot{\mathcal{A}}; \sigma)$ is a fuzzy $\diamond$-correct answer for $?A$, that is, for an arbitrary $\diamond$-model I for $\mathbb{P}$, we need to prove that

$$\dot{\mathcal{A}} \preceq I(A\sigma)$$

is satisfied. Since $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $?A$, then the atom A unifies with the head $H_1\theta\theta_1$ of a rule in $\mathbb{P}_r$, which comes from a set of critical rules in $\mathbb{P}$, $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \leq i \leq r\}$, with $\theta = \text{mgu}\{H_1, \ldots, H_r\}$. Therefore, the rule in $\mathbb{P}_r$ is

$$\begin{aligned}\langle H_1\theta\theta_1 \leftarrow @\big(&@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots,\\ &@_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r)\big), \top\rangle\end{aligned}$$

with $\theta_1 \in \mathcal{S}_g(H_1\theta)$, $\xi_1^i, \ldots, \xi_{n_i}^i \in \mathcal{S}_g(\mathcal{B}_i\theta\theta_1)$, for all $i \in \{1, \ldots, r\}$, and there exists a substitution ν such that $A\nu = H_1\theta\theta_1$. In addition, the body of the rule can be written as

$$@(@_{\sup}(\vartheta_1 \bar{\&}_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \bar{\&}_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))$$
$$= @' \left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]$$

where $B_1, \ldots, B_p$ are the atoms appearing in the bodies of the rules in $C_M^{H_1\theta}$, $@'$ is the composition of the symbols of operators that appear in the body of the $\diamond$-reductant, and $\eta_1^1, \ldots, \eta_{m_p}^p \in \{\xi_1^1, \ldots, \xi_{n_r}^r\}$. Furthermore, the fact that $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $?A$, also implies that there exist

$$\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p \in L$$

such that

$$\dot{\mathcal{A}} = \dot{@}' \left[\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p\right]$$

where $\left\langle \dot{\mathcal{A}}_j^i; id \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $?B_i\theta\theta_1\eta_j^i$, which follows from the sequence

$$\left\langle B_i\theta\theta_1\eta_j^i; id \right\rangle \rightsquigarrow_{\text{AS}} \ldots \overset{(1)}{\rightsquigarrow}_{\text{AS}} \left\langle \mathcal{A}_j^i; \nu_j^i \left[\mathcal{V}ar(B_i\theta\theta_1\eta_j^i)\right] \right\rangle \overset{(2)}{\rightsquigarrow}_{\text{IS}} \left\langle \dot{\mathcal{A}}_j^i; id \right\rangle$$

for all $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, m_i\}$. In (1), ν_j^i is the most general unifier between $B_i\theta\theta_1\eta_j^i$ and the head of a rule in $\mathbb{P}_r$, or it is $\perp$ in case $B_i\theta\theta_1\eta_j^i$ does not match with the head of any rule in $\mathbb{P}_r$; while (2) holds because the heads of the rules in $\mathbb{P}_r$ are ground and, consequently, the substitutions $\nu_i^j = id$, for all $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, m_i\}$. Consequently, the fuzzy $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ follows from the sequence

$$\begin{aligned}
\langle A; id \rangle &\rightsquigarrow_{\text{AS}} \left\langle \top \&@' \left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]\nu; \nu \right\rangle \\
&\overset{(*)}{=} \left\langle \top \&@' \left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]; \nu \right\rangle \\
&\rightsquigarrow_{\text{AS}} \\
&\vdots \\
&\rightsquigarrow_{\text{AS}} \left\langle @' \left[\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p\right]; \nu \left[\mathcal{V}ar(A)\right] \right\rangle \\
&\rightsquigarrow_{\text{IS}} \left\langle \dot{@}' \left[\dot{\mathcal{A}}_1^1, \ldots, \dot{\mathcal{A}}_{m_1}^1, \ldots, \dot{\mathcal{A}}_1^p, \ldots, \dot{\mathcal{A}}_{m_p}^p\right]; \nu \right\rangle
\end{aligned}$$

where in (*) we have taken into account that the body of the rule in $\mathbb{P}_r$ has no variables. In addition, as the length of computation of $\langle \mathcal{A}; \sigma \rangle$ is $k < n$, then every admissible $\mathbb{P}_r$-computed answer $\langle \mathcal{A}^i_j; id \rangle$ has a length of computation less than n. Therefore, from the induction hypothesis, $(\mathcal{A}^i_j; id)$ is a fuzzy $\diamond$-correct answer for $?B_i\theta\theta_1\eta^i_j$, and then, for the given $\diamond$-model I, the following inequality holds

$$\dot{\mathcal{A}}^i_j \preceq I(B_i\theta\theta_1\eta^i_j) \tag{8}$$

for all $i \in \{1, \ldots, p\}, j \in \{1, \ldots, m_i\}$. Now, it only remains to prove that $\dot{\mathcal{A}} \preceq I(A\nu)$, which follows from the following chain of inequalities.

$$\begin{aligned}
\dot{\mathcal{A}} &= \dot{@}'\left[\dot{\mathcal{A}}^1_1, \ldots, \dot{\mathcal{A}}^1_{m_1}, \ldots, \dot{\mathcal{A}}^p_1, \ldots, \dot{\mathcal{A}}^p_{m_p}\right] \\
&\overset{(1)}{\preceq} \dot{@}'\left[I(B_1\theta\theta_1\eta^1_1), \ldots, I(B_1\theta\theta_1\eta^1_{m_1}), \ldots, \dot{I}(B_p\theta\theta_1\eta^p_1), \ldots, I(B_p\theta\theta_1\eta^p_{m_p})\right] \\
&= \hat{I}\left(@'\left[B_1\theta\theta_1\eta^1_1, \ldots, B_1\theta\theta_1\eta^1_{m_1}, \ldots, B_p\theta\theta_1\eta^p_1, \ldots, B_p\theta\theta_1\eta^p_{m_p}\right]\right) \\
&= \hat{I}\left(@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi^1_1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi^1_{n_1}), \ldots, \right. \\
&\qquad \left. @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi^r_1, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi^r_{n_r}))\right) \\
&= \dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi^1_1), \ldots, \vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi^1_{n_1})), \ldots, \\
&\qquad \dot{@}_{\sup}(\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi^r_1), \ldots, \vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi^r_{n_r}))) \\
&\overset{(2)}{=} \sup\left\{\dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi^1_1), \ldots, \vartheta_1 \dot{\&}_1 I(\mathcal{B}_1\theta\theta_1\xi^1_{n_1})), \ldots, \right. \\
&\qquad \left. \dot{@}_{\sup}(\vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi^r_1), \ldots, \vartheta_r \dot{\&}_r I(\mathcal{B}_r\theta\theta_1\xi^r_{n_r})))\xi \mid \xi \in \mathcal{S}_g\right\} \\
&\overset{(3)}{=} T_{\mathbb{P}_r}(I)(H_1\theta\theta_1) \\
&\overset{(4)}{\preceq} I(H_1\theta\theta_1) \\
&\preceq I(A\nu)
\end{aligned}$$

where (1) follows from the monotonicity of $\dot{@}'$ and the induction hypothesis (Expression (8)); (2) from the fact that there is only one rule with head $H_1\theta\theta_1$ in $\mathbb{P}_r$ and there are no variables in its body; (3) from the definition of $T_{\mathbb{P}_r}(I)(H_1\theta\theta_1)$; and (4) follows from the fact that I is a $\diamond$-model of $\mathbb{P}$ and, from Proposition 6, I is a model of $\mathbb{P}_r$. Therefore, $(\dot{\mathcal{A}}; \nu)$ is a fuzzy $\diamond$-correct answer for $?A$ with respect to $\mathbb{P}$. □

Corollary 1. *Let $\mathbb{P}_r$ be the r-program of $\mathbb{P}$ and $\mathcal{Q}$ be a goal. Then, every fuzzy $\mathbb{P}_r$-computed answer for $\mathcal{Q}$ is a fuzzy $\diamond$-correct answer for $\mathbb{P}$ and $\mathcal{Q}$.*

PROOF. Consider a fuzzy $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ for $\mathcal{Q} = \mathcal{Q}[A_1, \ldots, A_n]$, where $A_1, \ldots, A_n$ are the atoms appearing in the formula $\mathcal{Q}$. Given a $\diamond$-model I for $\mathbb{P}$, we need to prove that $(\dot{\mathcal{A}}; \sigma)$ is a fuzzy $\diamond$-correct answer for $\mathcal{Q}$, that is, $\dot{\mathcal{A}} \preceq \hat{I}(\mathcal{Q}\sigma)$.

As $\left\langle \dot{\mathcal{A}}; \sigma \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $\mathcal{Q}$, then there exist fuzzy $\mathbb{P}_r$-computed answers

$$\left\langle \dot{\mathcal{A}}_1; \sigma_1 \right\rangle, \ldots, \left\langle \dot{\mathcal{A}}_n; \sigma_n \right\rangle$$

for $A_1, \ldots, A_n$, respectively, such that

$$\dot{\mathcal{A}} = \dot{\mathcal{Q}}[\dot{\mathcal{A}}_1, \ldots, \dot{\mathcal{A}}_n]$$

where $\dot{\mathcal{Q}}$ is the aggregator of the interpretations associated with the symbols of operators that appear in the formula $\mathcal{Q}$. In addition, the atoms from $\mathcal{Q}$ are considered to be standardized apart, and then $\sigma = \sigma_1 \circ \cdots \circ \sigma_n$. The following chain of inequalities is satisfied.

$$\begin{array}{rcl} \dot{\mathcal{A}} & = & \dot{\mathcal{Q}}[\dot{\mathcal{A}}_1, \ldots, \dot{\mathcal{A}}_n] \\ & \overset{(*)}{\preceq} & \dot{\mathcal{Q}}[\hat{I}(A_1\sigma), \ldots, \hat{I}(A_n\sigma)] \\ & = & \hat{I}\left(\mathcal{Q}[A_1, \ldots, A_n]\sigma\right) \\ & = & \hat{I}\left(\mathcal{Q}\sigma\right) \end{array}$$

where $(*)$ follows from the monotonicity of the operators appearing in $\dot{\mathcal{Q}}$ and Theorem 7. □

6. Completeness results

In this section, completeness theorems will be studied. For this, we need some previous results. Recall that a first-order language $L_1(\Omega, \mathcal{V}, \mathcal{P}, \mathcal{F})$ and a multi-adjoint lattice $(L, \preceq, \dot{\&}_1, \ldots, \dot{\&}_m)$, and a MALP $\mathbb{P}$ have been fixed.

Lemma 4. *Let $\mathcal{Q}$ be a goal, α be an ordinal such that $(T_{\mathbb{P}}^{\diamond})^{\alpha}(\vartriangle) = lfp(T_{\mathbb{P}}^{\diamond})$ and $\dot{\mathcal{A}} \in L$. The inequality $\dot{\mathcal{A}} \preceq \widehat{(T_{\mathbb{P}}^{\diamond})^{\alpha}(\vartriangle)}(\mathcal{Q}\theta)$ is satisfied if and only if $(\dot{\mathcal{A}}; \theta)$ is a fuzzy $\diamond$-correct answer for $\mathbb{P}$ and $\mathcal{Q}$.*

PROOF. Let $\mathcal{Q}$ be a goal. On the one hand, consider $\dot{\mathcal{A}} \in L$ satisfying that

$$\dot{\mathcal{A}} \preceq \widehat{(T_{\mathbb{P}}^{\diamond})^{\alpha}(\vartriangle)}(\mathcal{Q}\theta) \tag{9}$$

As $(T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle)$ is the least $\diamond$-model of $\mathbb{P}$ (Theorem 4), then

$$\widehat{(T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle)}(\mathcal{Q}\theta) \preceq \hat{I}(\mathcal{Q}\theta) \tag{10}$$

for all $\diamond$-model I of $\mathbb{P}$. Consequently, from Expressions (9) and (10), the following inequality arises.

$$\dot{\mathcal{A}} \preceq \hat{I}(\mathcal{Q}\theta)$$

for all $\diamond$-model $I \in \mathcal{I}_{\mathfrak{L}}$. Therefore, $(\dot{\mathcal{A}};\theta)$ is a fuzzy $\diamond$-correct answer for $\mathbb{P}$ and $\mathcal{Q}$.

On the other hand, consider a fuzzy $\diamond$-correct answer $(\dot{\mathcal{A}};\theta)$ for $\mathbb{P}$ and $\mathcal{Q}$. Then, the following inequality holds

$$\dot{\mathcal{A}} \preceq \hat{I}(\mathcal{Q}\theta)$$

for all $\diamond$-model $I \in \mathcal{I}_{\mathfrak{L}}$. Considering the particular case of the $\diamond$-model $(T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle$), we obtain that $\dot{\mathcal{A}} \preceq \widehat{(T_{\mathbb{P}}^{\diamond})^{\alpha}(\triangle)}(\mathcal{Q}\theta)$. □

Next, we will prove that each iteration $\langle (T_{\mathbb{P}}^{\diamond})^{n}(\triangle)(A); id\rangle$, for every $n \in \mathbb{N}$, is a fuzzy $\mathbb{P}_r$-computed answer for the query $?A$.

Proposition 8. *Let $\mathbb{P}_r$ be the r-program of $\mathbb{P}$ and A be an atom. Then, $\langle (T_{\mathbb{P}}^{\diamond})^{n}(\triangle)(A\nu); \nu\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $?A$, for every $n \in \mathbb{N}$, where ν is the most general unifier between A and an atom C, head of a rule in $\mathbb{P}_r$, or $\nu = id$, in case A does not match with the head of a rule of $\mathbb{P}_r$.*

Proof. Consider an atom A. First of all, if A does not unify with the head of any rule of $\mathbb{P}_r$, then $\langle (T_{\mathbb{P}}^{\diamond})^{n}(\triangle)(A\nu); \nu\rangle = \langle \bot; id\rangle$, which is always a fuzzy $\mathbb{P}_r$-computed answer for $?A$, for all $n \in \mathbb{N}$. Now, we will assume that A matches with the head of a rule of $\mathbb{P}_r$. Hence, there exists some ground atom $H_1\theta\theta_1$ head of a rule in $\mathbb{P}_r$, which comes from the set of critical rules in $\mathbb{P}$, $C_M^{H_1\theta} = \{\langle H_i \leftarrow_i \mathcal{B}_i; \vartheta_i\rangle \mid 1 \le i \le r\}$, with $\theta = \text{mgu}\{H_1, \ldots, H_r\}$. Therefore, the rule in $\mathbb{P}_r$ is

$$\begin{aligned}\big\langle H_1\theta\theta_1 \leftarrow @\big(&@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots,\\ &@_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r)\big), \top\big\rangle\end{aligned}$$

with $\theta_1 \in \mathcal{S}_g(H_1\theta)$, $\xi_1^i, \ldots, \xi_{n_i}^i \in \mathcal{S}_g(\mathcal{B}_i\theta\theta_1)$, for all $i \in \{1, \ldots, r\}$, and there exists a substitution ν such that $A\nu = H_1\theta\theta_1$. In addition, the body of the rule can be written as

$$\begin{aligned}&@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\\ &= @'\left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]\end{aligned}$$

where $B_1, \ldots, B_p$ are the atoms appearing in the bodies of the rules in $C_M^{H_1\theta}$, $@'$ is the composition of the symbols of operators that appear in the body of the $\diamond$-reductant, and $\eta_1^1, \ldots, \eta_{m_p}^p \in \{\xi_1^1, \ldots, \xi_{n_r}^r\}$.

We will prove the result by induction on the number of iterations n of $\widehat{T_{\mathbb{P}}^{\diamond}(\triangle)}$.

For $n = 1$, the following chain of equalities holds.

$$\begin{array}{rcl}
T_{\mathbb{P}}^{\diamond}(\triangle)(A\nu) & = & T_{\mathbb{P}}^{\diamond}(\triangle)(H_1\theta\theta_1) \\
& \overset{(1)}{=} & \dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 \hat{\triangle}(\mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \vartheta_1 \dot{\&}_1 \hat{\triangle}(\mathcal{B}_1\theta\theta_1\xi_{n_1}^1)), \ldots, \\
& & \quad \dot{@}_{\sup}(\vartheta_r \dot{\&}_r \hat{\triangle}(\mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \vartheta_r \dot{\&}_r \hat{\triangle}(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r))) \\
& = & \hat{\triangle}\left(@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, \right. \\
& & \quad \left. @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\right) \\
& = & \hat{\triangle}\left(@'\left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]\right) \\
& = & \dot{@}'[\hat{\triangle}(B_1\theta\theta_1\eta_1^1), \ldots, \hat{\triangle}(B_1\theta\theta_1\eta_{m_1}^1), \ldots, \hat{\triangle}(B_p\theta\theta_1\eta_1^p), \ldots, \hat{\triangle}(B_p\theta\theta_1\eta_{m_p}^p)]
\end{array}$$

where (1) follows from Definition 25. Notice that $\hat{\triangle}(B_i\theta\theta_1\eta_j^i) \in \{\bot, \vartheta_j^i\}$, where ϑ_j^i is the truth-value of some fact in $\mathbb{P}_r$ with head $B_i\theta\theta_1\eta_j^i$, for all $i \in \{1, \ldots, p\}, j \in \{1, \ldots, m_i\}$. Consequently, following Definition 18, applying admissible rules R1 for those ground atoms which are head of a fact in $\mathbb{P}_r$ or R2, we have that

$$\left\langle \dot{@}'[\hat{\triangle}(B_1\theta\theta_1\eta_1^1), \ldots, \hat{\triangle}(B_1\theta\theta_1\eta_{m_1}^1), \ldots, \hat{\triangle}(B_p\theta\theta_1\eta_1^p), \ldots, \hat{\triangle}(B_p\theta\theta_1\eta_{m_p}^p)]; \nu \right\rangle$$

is a fuzzy $\mathbb{P}_r$-computed answer for $?A$.

Assume that the result is true for $n = k \geq 1$, that is, given an atom Q, the pair $\left\langle \widehat{(T_{\mathbb{P}}^{\diamond})^k(\triangle)}(Q\sigma); \sigma \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $?Q$, where σ is a substitution such that $Q\sigma = H_1\theta\theta_1$, and $H_1\theta\theta_1$ is the head of a rule in $\mathbb{P}_r$. Now, we will prove that $\left\langle \widehat{(T_{\mathbb{P}}^{\diamond})^{k+1}(\triangle)}(A\nu); \nu \right\rangle$ is also a fuzzy $\mathbb{P}_r$-computed answer for $?A$, where ν is a substitution such that $A\nu = H_1\theta\theta_1$, and $H_1\theta\theta_1$ is the head of a rule in $\mathbb{P}_r$; or $\nu = id$, in case no head of rule in $\mathbb{P}_r$ unifies with A.

Since we want $\left\langle \widehat{(T_{\mathbb{P}}^{\diamond})^{k+1}(\triangle)}(A\nu); \nu \right\rangle$ to be a $\mathbb{P}_r$-computed answer for $?A$, we need to construct a sequence of formulae starting from $\langle A; id\rangle$ and

ending with $\left\langle \widehat{(T_{\mathbb{P}}^{\diamond})^{k+1}}(\triangle)(A\nu); \nu \right\rangle$. For this, we need to compute a $\mathbb{P}_r$-computed answer for each query $?B_i\theta\theta_1\eta_j^i$ in the body of the rule of $\mathbb{P}_r$, for all $i \in \{1,\ldots,p\}, j \in \{1,\ldots,m_i\}$. From the induction hypothesis, $\left\langle (T_{\mathbb{P}}^{\diamond})^k(\triangle)(B_i\theta\theta_1\eta_j^i); id \right\rangle$ is a $\mathbb{P}_r$-computed answer for $?B_i\theta\theta_1\eta_j^i$, where the substitution is id since $B_i\theta\theta_1\eta_j^i$ is ground, for all $i \in \{1,\ldots,p\}, j \in \{1,\ldots,m_i\}$. In order to simplify the notation, we will denote $(T_{\mathbb{P}}^{\diamond})^k(\triangle)(B_i\theta\theta_1\eta_j^i) = \beta_j^i$, for all $i \in \{1,\ldots,p\}, j \in \{1,\ldots,m_i\}$. A $\mathbb{P}_r$-computed answer is obtained from the following chain of equalities.

$$
\begin{array}{rcl}
\langle A; id\rangle & \rightsquigarrow_{\text{AS}} & \left\langle \top \&@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, \right. \\
& & \qquad \left. @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\nu; \nu \right\rangle \\
& = & \left\langle @'\left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]; \nu \right\rangle \\
& \rightsquigarrow_{\text{AS}} & \\
& \vdots & \\
& \rightsquigarrow_{\text{AS}} & \left\langle @'[\beta_1^1, \ldots, \beta_{m_1}^1, \ldots, \beta_1^p, \ldots, \beta_{m_p}^p]; \nu \right\rangle \\
& \rightsquigarrow_{\text{IS}} & \left\langle \dot{@}'[\beta_1^1, \ldots, \beta_{m_1}^1, \ldots, \beta_1^p, \ldots, \beta_{m_p}^p]; \nu \right\rangle \\
& = & \left\langle \dot{\mathcal{A}}; \nu \right\rangle
\end{array}
$$

Finally, the following chain of equalities is reached.

$$
\begin{array}{rcl}
\left\langle \dot{\mathcal{A}}; \nu \right\rangle & = & \dot{@}'[\beta_1^1, \ldots, \beta_{m_1}^1, \ldots, \beta_1^p, \ldots, \beta_{m_p}^p] \\
& = & \dot{@}'[(T_{\mathbb{P}}^{\diamond})^k(\triangle)(B_1\theta\theta_1\eta_1^1), \ldots, (T_{\mathbb{P}}^{\diamond})^k(\triangle)(B_1\theta\theta_1\eta_{m_1}^1), \ldots, \\
& & \quad (T_{\mathbb{P}}^{\diamond})^k(\triangle)(B_p\theta\theta_1\eta_1^p), \ldots, (T_{\mathbb{P}}^{\diamond})^k(\triangle)(B_p\theta\theta_1\eta_{m_p}^p)] \\
& = & \widehat{(T_{\mathbb{P}}^{\diamond})^k}(\triangle)\left(@'\left[B_1\theta\theta_1\eta_1^1, \ldots, B_1\theta\theta_1\eta_{m_1}^1, \ldots, B_p\theta\theta_1\eta_1^p, \ldots, B_p\theta\theta_1\eta_{m_p}^p\right]\right) \\
& = & \widehat{(T_{\mathbb{P}}^{\diamond})^k}(\triangle)\left(@(@_{\sup}(\vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_1^1, \ldots, \vartheta_1 \&_1 \mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, \right. \\
& & \qquad \left. @_{\sup}(\vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_1^r, \ldots, \vartheta_r \&_r \mathcal{B}_r\theta\theta_1\xi_{n_r}^r))\right) \\
& = & \dot{@}(\dot{@}_{\sup}(\vartheta_1 \dot{\&}_1 \widehat{(T_{\mathbb{P}}^{\diamond})^k}(\triangle)(\mathcal{B}_1\theta\theta_1\xi_1^1), \ldots, \vartheta_1 \dot{\&}_1 \widehat{(T_{\mathbb{P}}^{\diamond})^k}(\triangle)(\mathcal{B}_1\theta\theta_1\xi_{n_1}^1), \ldots, \\
& & \quad \dot{@}_{\sup}(\vartheta_r \dot{\&}_r \widehat{(T_{\mathbb{P}}^{\diamond})^k}(\triangle)(\mathcal{B}_r\theta\theta_1\xi_1^r), \ldots, \vartheta_r \dot{\&}_r \widehat{(T_{\mathbb{P}}^{\diamond})^k}(\triangle)(\mathcal{B}_r\theta\theta_1\xi_{n_r}^r)))) \\
& \overset{(*)}{=} & (T_{\mathbb{P}}^{\diamond})^{k+1}(\triangle)(H_1\theta\theta_1) \\
& = & (T_{\mathbb{P}}^{\diamond})^{k+1}(\triangle)(A\nu)
\end{array}
$$

where (*) follows from Definition 25. As a consequence,

$$\left\langle (T_{\mathbb{P}}^{\diamond})^{k+1}(\triangle)(A\nu); \nu \right\rangle = \left\langle \dot{@}'[\beta_1^1, \ldots, \beta_{m_1}^1, \ldots, \beta_1^p, \ldots, \beta_{m_p}^p]; \nu \right\rangle$$

is a fuzzy $\mathbb{P}_r$-computed answer for $?A$. □

Corollary 2. *Let $\mathbb{P}_r$ be the r-program of $\mathbb{P}$ and $\mathcal{Q}$ be a goal. Then, for every $n \in \mathbb{N}$, there exists a substitution ν such that $\left\langle \widehat{(T_{\mathbb{P}}^{\diamond})^n(\triangle)}(\mathcal{Q}\nu); \nu \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $\mathcal{Q}$.*

Proof. Consider an arbitrary $n \in \mathbb{N}$ and a goal $\mathcal{Q} = \mathcal{Q}[A_1, \ldots, A_r]$, where $A_1, \ldots, A_r$ are the atoms appearing in the formula $\mathcal{Q}$. A fuzzy $\mathbb{P}_r$-computed answer for $\mathcal{Q}$ follows from the sequence

$$\langle \mathcal{Q}; id \rangle \;=\; \langle \mathcal{Q}[A_1, \ldots, A_r]; id \rangle \rightsquigarrow_{\text{AS}} \ldots \rightsquigarrow_{\text{IS}} \left\langle \dot{\mathcal{Q}}[\beta_1, \ldots, \beta_r]; \nu_1 \circ \cdots \circ \nu_r \right\rangle$$

where $\langle \beta_i; \nu_i \rangle = \langle (T_{\mathbb{P}}^{\diamond})^n(\triangle)(A_i\nu_i); \nu_i \rangle$, which is a fuzzy $\mathbb{P}_r$-computed answer for $?A_i$ by Proposition 8, for all $i \in \{1, \ldots, r\}$. We are considering the atoms $A_1, \ldots, A_r$ to be standardized apart and, hence, we can consider $\nu = \nu_1 \circ \cdots \circ \nu_r$. Now, the following chain of equalities arises.

$$\begin{aligned} \dot{\mathcal{Q}}[\beta_1, \ldots, \beta_r] &= \dot{\mathcal{Q}}[(T_{\mathbb{P}}^{\diamond})^n(\triangle)(A_1\nu), \ldots, (T_{\mathbb{P}}^{\diamond})^n(\triangle)(A_r\nu)] \\ &= \widehat{(T_{\mathbb{P}}^{\diamond})^n(\triangle)}\,(\mathcal{Q}[A_1, \ldots, A_r]\nu) \\ &= \widehat{(T_{\mathbb{P}}^{\diamond})^n(\triangle)}\,(\mathcal{Q}\nu) \end{aligned}$$

Therefore, $\left\langle \widehat{(T_{\mathbb{P}}^{\diamond})^n(\triangle)}\,(\mathcal{Q}\nu)\,; \nu \right\rangle$ is a fuzzy $\mathbb{P}_r$-computed answer for $\mathcal{Q}$. □

Finally, before proving the completeness theorems, we need the following technical result.

Proposition 9. *It is satisfied that $(T_{\mathbb{P}}^{\diamond})^n(\triangle) \sqsubseteq (T_{\mathbb{P}}^{\diamond})^{n+1}(\triangle)$, for all $n \in \mathbb{N}$.*

Proof. We will prove the result by induction over n. For $n = 1$, it is clear that $\triangle \sqsubseteq T_{\mathbb{P}}^{\diamond}(\triangle)$ always holds and, as $T_{\mathbb{P}}^{\diamond}$ is monotonic, then $T_{\mathbb{P}}^{\diamond}(\triangle) \sqsubseteq T_{\mathbb{P}}^{\diamond}\,(T_{\mathbb{P}}^{\diamond}(\triangle)) = (T_{\mathbb{P}}^{\diamond})^2(\triangle)$ is satisfied.

Now, assume that $(T_{\mathbb{P}}^{\diamond})^{n-1}\,(\triangle) \sqsubseteq (T_{\mathbb{P}}^{\diamond})^n(\triangle)$ is true for $1 < n$. As $T_{\mathbb{P}}^{\diamond}$ is monotonic, then

$$(T_{\mathbb{P}}^{\diamond})^n(\triangle) = T_{\mathbb{P}}^{\diamond}\left((T_{\mathbb{P}}^{\diamond})^{n-1}\,(\triangle)\right) \sqsubseteq T_{\mathbb{P}}^{\diamond}\,((T_{\mathbb{P}}^{\diamond})^n(\triangle)) = (T_{\mathbb{P}}^{\diamond})^{n+1}(\triangle)$$

and the result is obtained. □

The first completeness result shows that given a certain $\diamond$-correct answer λ, we can find a $\mathbb{P}_r$-computed answer as close as we want to λ. This result requires the continuity of the $T_{\mathbb{P}}^{\diamond}$ operator. As mentioned in [22], when we are working on a practical and real system, it is essential that the operator $T_{\mathbb{P}}^{\diamond}$ be Scott-continuous[2], or just continuous, in order to ensure not only the existence of such computational model, but also that it can be computationally reached in numerable number of iterations ω. Hence, this requirement is not a strong restriction when we are considering real cases.

Theorem 8. *Let A be an atom unifying with at least one rule in $\mathbb{P}_r$. If $T_{\mathbb{P}}^{\diamond}$ is continuous, then for each fuzzy $\diamond$-correct answer $(\lambda;\theta)$ for $\mathbb{P}$ and $?A$, and for all $\varepsilon \prec \lambda$, there is a $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}};\sigma \right\rangle$ such that $\varepsilon \prec \dot{\mathcal{A}}$.*

Proof. Given an atom A unifying with at least one rule in $\mathbb{P}_r$, since $(\lambda;\theta)$ is a fuzzy $\diamond$-correct answer for $?A$, then from the continuity of $T_{\mathbb{P}}^{\diamond}$, Theorem 4 and Lemma 4, we have that $\lambda \preceq \widehat{(T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)}(A\theta)$. Taking into account the continuity of $T_{\mathbb{P}}^{\diamond}$ and Proposition 9, we reached that

$$\varepsilon \prec \lambda \preceq \widehat{(T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)}(A\theta) \overset{(*)}{\preceq} (T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)(A\theta\nu) = \sup\left\{(T_{\mathbb{P}}^{\diamond})^{n}(\triangle)(A\theta\nu) \,|\, n \in \mathbb{N}\right\}$$

where (*) follows from Lemma 1, and ν is a substitution such that $A\theta\nu = H_1\theta'\theta'_1$, with $H_1\theta'\theta'_1$ is the head of a rule in $\mathbb{P}_r$. Therefore, by Proposition 9, since $X = \{(T_{\mathbb{P}}^{\diamond})^{n}(\triangle)(A\theta\nu) \,|\, n \in \mathbb{N}\}$ is an increasing sequence of elements in L with limit greater or equal than λ, then exists some $\dot{\mathcal{A}} \in X$ (or equivalently $n_0 \in \mathbb{N}$), that is, $\dot{\mathcal{A}} = (T_{\mathbb{P}}^{\diamond})^{n_0}(\triangle)(A\theta\nu)$, such that

$$\varepsilon \prec \dot{\mathcal{A}} \preceq (T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)(A\theta\nu)$$

which finishes the proof because, by Proposition 8, $\langle (T_{\mathbb{P}}^{\diamond})^{n_0}(\triangle)(A\theta\nu);\nu\rangle$ is a $\mathbb{P}_r$-computed answer. □

Finally, following quasi-completeness theorem is proved.

Theorem 9. *Let A be an atom unifying with at least one rule in $\mathbb{P}_r$. If $T_{\mathbb{P}}^{\diamond}$ is continuous, then for each fuzzy $\diamond$-correct answer $(\lambda;\theta)$ for $\mathbb{P}$ and a query $?A$, there is a sequence of elements $\lambda_n \in L$, with $\lambda \preceq \sup \lambda_n$, such that, for any $n_0 \in \mathbb{N}$, there exists a $\mathbb{P}_r$-computed answer $\left\langle \dot{\mathcal{A}};\nu \right\rangle$ verifying $\lambda_{n_0} \preceq \dot{\mathcal{A}}$.*

[2]We recall that an operator T is *Scott-continuous* if it preserves suprema of directed sets, i.e., given a directed set $X \subseteq L$ it is satisfied that $T(\sup X) = \sup\{T(x) \,|\, x \in X\}$ [21].

PROOF. Consider an atom A unifying with at least one rule in $\mathbb{P}_r$, a fuzzy $\diamond$-correct answer $(\lambda;\theta)$ for $?A$ and the sequence $\lambda_n = \widehat{(T_{\mathbb{P}}^{\diamond})^n(\triangle)}(A\theta\nu)$, for all $n \in \mathbb{N}$. From Lemma 4 and the continuity of $T_{\mathbb{P}}^{\diamond}$, it is satisfied that

$$\lambda \preceq \widehat{(T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)}(A\theta) \overset{(*)}{\preceq} (T_{\mathbb{P}}^{\diamond})^{\omega}(\triangle)(A\theta\nu) = \sup\{(T_{\mathbb{P}}^{\diamond})^n(\triangle)(A\theta\nu) \mid n \in \mathbb{N}\} = \sup \lambda_n$$

where (*) follows from Lemma 1, and ν is a substitution such that $A\theta\nu = H_1\theta'\theta'_1$, being $H_1\theta'\theta'_1$ the head of a rule in $\mathbb{P}_r$.

Now, given $n_0 \in \mathbb{N}$, we consider the value $\dot{\mathcal{A}} = (T_{\mathbb{P}}^{\diamond})^{n_0}(\triangle)(A\theta\nu)$, such that $\langle (T_{\mathbb{P}}^{\diamond})^{n_0}(\triangle)(A\theta\nu); \nu\rangle$ is a $\mathbb{P}_r$-computed answer for $?A$ by Proposition 8. □

7. Conclusions and future work

In this paper, we have presented a new operational semantics for an arbitrary framework based on a general monotonic immediate consequences operator $T_{\mathbb{P}}^{\diamond}$, generalizing the specific cases introduced in [26, 25, 27]. For this, the definitions of correct answer and computed answer for a given query have been adapted, and a new and more versatile definition of reductant has been introduced, with the aim of computationally obtaining the correct answer that provides the most useful information, which can be done through a new logic program $\mathbb{P}_r$, composed of reductants, which is semantically equivalent to the original program (that is, the information is preserved) and simplify the entire computation process. Finally, the soundness of the procedure and some completeness results have been studied, allowing to be applied on intelligent systems for decision-support, obtaining answers to queries to the program.

As future work, this flexible operational semantics will be applied to real datasets with imperfect and low quality data. From this practical study, we will also analyze the necessity of defining new specific definitions of the operator $T_{\mathbb{P}}^{\diamond}$ and we will inspect new properties. Furthermore, a deeper research on the relationships that can be established between the new approaches and those already introduced in [26, 25, 27] will be done.

References

[1] J. Almendros-Jimenez, A. Becerra-Terón, G. Moreno, and J. Riaza. Tuning fuzzy SPARQL queries in a fuzzy logic programming environment. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7, 06 2019.

[2] J. M. Almendros-Jiménez, M. Bofill, A. Luna-Tedesqui, G. Moreno, C. Vázquez, and M. Villaret. Fuzzy XPath for the automatic search of fuzzy formulae models. In C. Beierle and A. Dekhtyar, editors, *Scalable Uncertainty Management*, pages 385–398, Cham, 2015. Springer International Publishing.

[3] J. M. Almendros-Jiménez, A. Becerra-Terón, and G. Moreno. Fuzzy queries of social networks with FSA-SPARQL. *Expert Systems with Applications*, 113:128–146, 2018.

[4] A. R. Bradley and Z. Manna. *First-Order Logic*, pages 35–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[5] E. Çakir and Z. Ulukan. A fuzzy linguistic programming for sustainable ecotourism activities. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 01 2020.

[6] M. E. Cornejo, J. Medina, and E. Ramírez-Poussa. Multi-adjoint algebras versus non-commutative residuated structures. *International Journal of Approximate Reasoning*, 66:119–138, 2015.

[7] M. E. Cornejo, J. Medina, and E. Ramírez-Poussa. Algebraic structure and characterization of adjoint triples. *Fuzzy Sets and Systems*, 425:117–139, 2021. Mathematics.

[8] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201?215, July 1960.

[9] M. v. Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[10] P. C. Gilmore. A proof method for quantification theory: its justification and realization. *IBM J. Res. Dev.*, 4(1):28?35, Jan. 1960.

[11] J. Herbrand. Investigations in proof theory. 1930. *English translation in vH67*, 1971.

[12] P. Julián-Iranzo, J. Medina, and M. Ojeda-Aciego. On reductants in the framework of multi-adjoint logic programming. *Fuzzy Sets and Systems*, 317:27 – 43, 2017.

[13] P. Julián-Iranzo and C. Rubio-Manzano. Bousi˜Prolog - A fuzzy logic programming language for modeling vague knowledge and approximate reasoning. In J. Filipe and J. Kacprzyk, editors, *ICFC-ICNC 2010 - Proceedings of the International Conference on Fuzzy Computation and International Conference on Neural Computation, [parts of the International Joint Conference on Computational Intelligence IJCCI 2010], Valencia, Spain, October 24-26, 2010*, pages 93–98. SciTePress, 2010.

[14] P. Julián-Iranzo, G. Moreno, and J. A. Riaza. The fuzzy logic programming language FASILL: Design and implementation. *International Journal of Approximate Reasoning*, 125:139–168, 2020.

[15] P. Julián-Iranzo and F. Sáenz-Pérez. Bousi-Prolog: Design and implementation of a proximity-based fuzzy logic programming language. *Expert Systems with Applications*, 213:118858, 2023.

[16] A. Kishor, A. K. Singh, and N. R. Pal. Orness measure of owa operators: A new approach. *IEEE Transactions on Fuzzy Systems*, 22(4):1039–1045, 2014.

[17] R. Kowalski. *Logic for Problem Solving*. 01 1979.

[18] I. Lizasoain and C. Moreno. Owa operators defined on complete lattices. *Fuzzy Sets and Systems*, 224:36 – 52, 2013. Theme: Aggregation functions and implications.

[19] J. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.

[20] M. Mahout, R. P. Carlson, L. Simon, and S. Peres. Logic programming-based minimal cut sets reveal consortium-level therapeutic targets for chronic wound infections.

npj Systems Biology and Applications, 10(1):34, 2024.

[21] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. In *Logic Programming and Non-Monotonic Reasoning, LPNMR'01*, pages 351–364. Lecture Notes in Artificial Intelligence 2173, 2001.

[22] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programming. In *Progress in Artificial Intelligence, EPIA'01*, pages 290–297. Lecture Notes in Artificial Intelligence 2258, 2001.

[23] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. In *Proc. EUSFLAT Conference in Fuzzy Logic and Technology*, pages 273–276, 2001.

[24] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 146:43–62, 2004.

[25] J. Medina and J. A. Torné. *Fuzzy Logic Programming with Generalized Quantifiers*, pages 17–23. Springer International Publishing, Cham, 2023.

[26] J. Medina and J. A. Torné-Zambrano. Immediate consequences operator on generalized quantifiers. *Fuzzy Sets and Systems*, 2022.

[27] J. Medina and J. A. Torné-Zambrano. Immediate consequences operator on through ordered weighted average operators. *Iranian Journal of Fuzzy Sets*, 20(5):1–18, 2023.

[28] J. Medina and J. A. Torné-Zambrano. Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers. *Applied and Computational Mathematics*, 23(4):520–540, 2024.

[29] J. Medina and J. A. Torné-Zambrano. Revisiting immediate consequences operator on first-order logic programming. *Lecture Notes in Networks and Systems*, 1175, 2024. In press.

[30] G. Moreno, J. Pérez, and J. A. Riaza. Fuzzy logic programming for tuning neural networks. In P. Fodor, M. Montali, D. Calvanese, and D. Roman, editors, *Rules and Reasoning*, pages 190–197, Cham, 2019. Springer International Publishing.

[31] S. Morishita. A unified approach to semantics of multi-valued logic programs. Technical Report RT 5006, IBM Tokyo, 1990.

[32] J. Pavelka. On fuzzy logic i many-valued rules of inference. *Mathematical Logic Quarterly*, 25(3-6):45–52, 1979.

[33] D. PRAWITZ. An improved proof procedure. *Theoria*, 26(2):102–139, 1960.

[34] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23?41, Jan. 1965.

[35] F. P. Romero, P. Julián-Iranzo, A. Soto, M. Ferreira-Satler, and J. Gallardo-Casero. Classifying unlabeled short texts using a fuzzy declarative approach. *Language Resources and Evaluation*, 47(1):151–178, mar 2013.

[36] C. Rubio-Manzano and P. J. Iranzo. Incorporation of abstraction capability in a logic-based framework by using proximity relations. *Journal of Intelligent & Fuzzy Systems*, 29:1671–1683, 2015.

[37] A. Srinivasan, A. Baskar, T. Dash, and D. Shah. Composition of relational features with an application to explaining black-box predictors. *Machine Learning*, 113(3):1091–1132, 2024.

[38] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955.

[39] L. Tomás, C. Vázquez Pérez-Íñigo, J. Tordsson, and G. Moreno. Reducing noisy-

neighbor impact with a fuzzy affinity-aware scheduler. In *2015 International Conference on Cloud and Autonomic Computing*, pages 33–44, 09 2015.

[40] P. Vojtáš. Fuzzy logic programming. *Fuzzy sets and systems*, 124(3):361–370, 2001.

[41] R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1988.

[42] R. R. Yager. Families of OWA operators. *Fuzzy Sets and Systems*, 59(2):125 – 148, 1993.

[43] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En este capítulo final estableceremos las conclusiones a las que hemos llegado entre todos los artículos publicados y que conforman el presente trabajo.

El objetivo principal de la tesis ha sido introducir un paradigma de programación lógica multiadjunta más flexible, con el que poder diseñar y modelar sistemas expertos de conocimiento a partir de bases de datos con incertidumbre. Para ello, hemos extendido el operador de consecuencias inmediatas, esencial para definir la semántica de un programa lógico, a partir de, por un lado, cuantificadores generalizados [129] y, por otro lado, de operadores de promedio ponderado ordenado (los operadores OWA) [130]. Estas innovadoras aportaciones cumplen propiedades relevantes y garantizan una mayor versatilidad en el tratamiento de sistemas expertos con una mala calidad de los datos. En concreto, las principales conclusiones alcanzadas en cada uno de los artículos las presentamos a continuación.

- **Conclusiones alcanzadas en *Immediate consequences operator on generalized quantifiers***

  Tras exponer la problemática asociada a la posible presencia de una mala calidad de datos en las bases de datos y sus implicaciones en la definición del operador de consecuencias inmediatas a partir del supremo, hemos propuesto una nueva definición del operador de consecuencias inmediatas basada en los cuantificadores generalizados, en [129]. Esta versión cuantificada del operador no solo es significativamente más flexible que la versión original, sino que también demuestra una mayor capacidad para generar mejores modelos a partir de conjuntos de datos reales. Además, esta definición está basada en una familia de medidas difusas, de manera que se puede elegir la medida difusa más conveniente para adaptarse a cada caso particular que representan los símbolos proposicionales.

  Esta generalización verifica propiedades muy interesantes como la monotonía, la cual garantiza que si el factor de confianza de los datos iniciales se incrementa, entonces el valor obtenido para las conclusiones también va a ser mayor. Posteriormente, partiendo del análisis de un caso particular en el que el operador de consecuencias inmediatas cuantificada se queda con el segundo valor más alto, hemos estudiado una semántica de punto fijo adaptada a este entorno más flexible. En concreto, se ha introducido el concepto de M-modelo, dependiente de la familia de medidas difusas M que estemos considerando, y se ha podido caracterizar el menor M-modelo del programa, que es el que contiene información más precisa del programa, como el menor punto fijo del operador de consecuencias inmediatas cuantificado.

  Finalmente, hemos abordado la problemática de elegir una medida difusa para cada símbolo proposicional en base a dos requisitos. Por un lado, que elimine la posible presencia de ruido en los datos, con

lo que la medida difusa tiene que penalizar, aunque sea ligeramente, el valor o los valores más altos en la agregación. Por otro lado, que nos permita quedarnos con valores cercanos al que aportaría el supremo. Para medir este último requisito, hemos adaptado la noción de índice de optimismo (*orness* en su notación inglesa), introducida en el entorno de los operadores de agregación promedio ponderados, a este nuevo escenario. Por tanto, atendiendo a estas condiciones, podemos elegir una familia de medidas difusas conveniente para flexibilizar el operador de consecuencias inmediatas. A pesar de haber introducido una nueva definición del operador de consecuencias inmediatas que ayuda a descartar la posible presencia de datos disonantes en los datos, quedó por estudiar la continuidad de esta nueva aproximación, lo que se realizó en el siguiente artículo.

- **Conclusiones alcanzadas en *Computational reaching of quantified consequences from imperfect initial data***

  En este artículo, continuando el trabajo iniciado en [129], hemos presentado condiciones suficientes y necesarias para asegurar la continuidad del operador de consecuencias inmediatas cuantificado $T_{\mathbb{P}}^{\mathrm{M}}$. Estos resultados garantizan que el menor M-modelo del programa puede ser alcanzado en una cantidad numerable de iteraciones, por lo que se puede obtener a través de una semántica operacional.

  Asimismo, en este artículo hemos introducido un caso práctico en el cual hemos analizado las implicaciones derivadas de la falta de continuidad del operador de consecuencias inmediatas cuantificado. Específicamente, hemos observado cómo pequeñas variaciones en los datos de entradas pueden generar cambios significativos en las conclusiones obtenidas, subrayando aún más la importancia de garantizar la continuidad de dicho operador.

- **Conclusiones alcanzadas en *Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers***

  Una vez que hemos introducido una semántica de punto fijo para la nueva aproximación del operador de consecuencias inmediatas, y con el objetivo de poder desarrollar un modelo computacional válido, en el artículo [131] hemos presentado una nueva semántica operacional, con el fin de definir un procedimiento de razonamiento automático que facilite la obtención de deducciones e información del sistema modelado, así como de proporcionar respuestas a las consultas planteadas al mismo.

  Como punto de partida, se ha introducido una nueva definición de respuesta correcta para poder trabajar en este ambiente más versátil, formalizada a través del concepto de M-respuesta correcta, la cual depende de la familia de medidas difusas M que se considere. Además, se ha demostrado que la M-respuesta correcta máxima, que es la que proporciona la solución más precisa, puede ser obtenida iterativamente a partir del operador de consecuencias inmediatas cuantificado, ejemplificando nuevamente la importancia de garantizar la continuidad de dicho operador.

  Posteriormente, se ha aportado una nueva definición de reductante basada en cuantificadores generalizados. Asimismo, se ha introducido un nuevo programa lógico, programa reducido por M-reductantes, para el que hemos demostrado que es semánticamente equivalente al programa original. Esta ventaja nos permite poder trabajar exclusivamente con este programa que, además, facilita el cálculo de la $\mathbb{P}_r$-respuesta calculada máxima. Cabe destacar que, sabiendo que los reductantes se introducen originalmente para poder obtener computacionalmente la respuesta correcta máxima en situaciones donde es-

to no es posible, este análisis permite extrapolar los resultados satisfactorios alcanzados a otros contextos.

La noción de $\mathbb{P}_r$-respuesta calculada ha sido introducida, y hemos demostrado que la nueva semántica operacional es correcta, lo cual nos permite asegurar, para una pregunta dada, que todas las $\mathbb{P}_r$-respuestas calculadas son M-respuestas correctas. Algunos resultados de completitud han sido estudiados para el nuevo procedimiento, donde el requisito de la continuidad del operador de consecuencias inmediatas cuantificado es esencial.

Finalmente, hemos abordado el estudio de la $\mathbb{P}_r$-respuesta calculada máxima, aportando un procedimiento, basado en el uso de las reglas admisibles y del programa reducido, que nos permite obtenerla en caso de que exista. Aunque todavía no se han aportado condiciones para asegurar la existencia de dicha $\mathbb{P}_r$-respuesta calculada máxima, en caso de que exista, sí hemos demostrado que se trata del menor punto fijo del operador de consecuencias cuantificado que, además, se corresponde con la M-respuesta correcta máxima. En consecuencia, se ha demostrado que la respuesta que más información aporta se puede obtener a través de esta nueva semántica operacional.

- **Conclusiones alcanzadas en *Immediate consequences operator through ordered weighted average operators***

  Basándonos en las ventajas de los operadores de agregación ponderados para la fusión de elementos con distintas relaciones de importancia, en el artículo [130] hemos presentado una definición alternativa del operador de consecuencias inmediatas a partir de los operadores de promedio ponderado ordenado. Específicamente, se ha introducido el concepto de operador de agregación ponderado de consecuencias inmediatas. Esta nueva definición ha probado ser también útil para reducir la posible presencia de ruido en los da-

tos que se utilizan para modelar las reglas lógicas de un programa. Además, el operador de agregación ponderado de consecuencias inmediatas se construye a partir de una familia de vectores de pesos, lo que nos permite seleccionar el vector de pesos más adecuado para cada caso específico.

Asimismo, se ha demostrado que el nuevo operador es monótono y se han estudiado condiciones suficientes y necesarias para asegurar su continuidad, garantizando, bajo las condiciones establecidas, que el menor punto fijo del operador sea alcanzable en una cantidad numerable de iteraciones.

El siguiente paso natural ha sido estudiar una semántica de punto fijo para esta nueva aproximación. Para ello, se ha introducido la definición de $\mathcal{W}$-modelo, adaptada a este ambiente de reducir el impacto de los datos disruptivos. Se ha demostrado que el menor $\mathcal{W}$-modelo del programa se corresponde con el menor punto fijo del operador de agregación ponderado de consecuencias inmediatas, y que además se puede obtener computacionalmente. La continuidad de dicho operador asegura este hecho.

Por último, con respecto a la selección de un vector de pesos conveniente, se ha adaptado la noción de índice de optimismo a este nuevo contexto, con el objetivo de poder elegir vectores de pesos con los que obtener operadores cercanos al supremo.

- **Conclusiones alcanzadas en *Operational semantics on arbitrary first-order fuzzy logic programming***

  En este artículo [128] hemos presentado una nueva semántica operacional para un contexto arbitrario de programación lógica de primer orden multiadjunta, basada en un operador de consecuencias inmediatas monótono general $T^{\diamond}_{\mathbb{P}}$, que abarca los casos propuestos en [129, 131, 130].

Se han adaptado las definiciones de respuesta correcta y respuesta calculada al nuevo marco de trabajo. Además, se ha presentado una nueva definición de reductante, más flexible, y se ha introducido un nuevo programa lógico formado por reductantes, que preserva la información del programa de partida, y que permite adaptar la noción de respuesta calculada al nuevo ambiente, a la vez que simplifica todo el proceso de computación. Finalmente, se ha estudiado la corrección de la nueva semántica operacional, junto con algunos resultados sobre la completitud de la misma.

8.2. Trabajo futuro

Como conclusión, se abordarán las líneas de trabajo futuro que constituyen la continuación natural de la presente tesis doctoral.

En primer lugar, fijémonos que las generalizaciones del operador de consecuencias inmediatas propuestas, tanto a través de los cuantificadores generalizados como de los operadores de promedio ponderado ordenado, han sido presentadas en el retículo completo $[0, 1]$ con un orden total. Por lo tanto, uno de los primeros objetivos más inmediatos consiste en extender estos resultados y definiciones a un retículo completo más general $\langle L, \preceq \rangle$. Para esto, usaremos, por ejemplo, las extensiones de los operadores de agregación ponderados introducidos por Yager [209], dados para un retículo arbitrario en [117].

Además de generalizar estos resultados a un retículo arbitrario, también se continuará con la ampliación de este marco de programación lógica multiadjunta flexible a un contexto de primer orden, profundizando en los casos particulares de los operadores basados en cuantificadores generalizados, en operadores OWA y en otras posibles extensiones.

Una vez completado este estudio, se pretende analizar otras extensio-

nes de dicho operador, basadas en otros operadores formales de agregación y en teorías alternativas de los cuantificadores generalizados, como la teoría de cuantificadores intermedios propuesta en [142].

Otra de las metas que nos proponemos realizar a corto plazo es el estudio de condiciones que permitan asegurar la obtención de la $\mathbb{P}_r$-respuesta calculada máxima. Nuestro objetivo abarcará igualmente el esfuerzo por establecer una relación entre dicha obtención y los resultados de terminación del operador de consecuencias inmediatas [51, 42].

Asimismo, hemos podido comprobar que no todas las medidas difusas ni todos los vectores de pesos son convenientes para generalizar el operador de consecuencias inmediatas. En consecuencia, se estudiarán condiciones que nos permitan asegurar que una cierta generalización del operador es adecuada. Todo ello se relaciona también con la necesidad de realizar un estudio que nos permita relacionar las dos aproximaciones del operador de consecuencias inmediatas presentadas en este trabajo.

Finalmente, entre los objetivos a largo plazo, se incluye la implementación de los resultados obtenidos en programas como Bousi$\sim$Prolog o FASILL, donde, además, se contemplará el uso de bases de datos reales y escenarios actuales, en los cuales el tratamiento de la incertidumbre presente en los datos será fundamental para alcanzar aproximaciones óptimas.

Anexos
Otras publicaciones relacionadas

Anexo A

Fuzzy Logic Programming with Generalized Quantifiers

Fuzzy Logic Programming with Generalized Quantifiers

Jesús Medina and José Antonio Torné

Abstract This paper introduces a new definition of the immediate consequences operator considered in many fuzzy logic programming frameworks, focused on weakening the existential feature of the supremum operator. This new definition has been possible taking advantage of the notion of generalized quantifiers, which provides weaker quantifiers than the existential one.

Keywords Logic programming · Immediate consequences operator · Generalized quantifiers

1 Introduction

Fixed-point semantics is one of the most useful mechanisms for obtaining information from datasets modeled by logic programs [10–13, 15–17]. In the monotonic case, it is characterized through the least fixed-point of the immediate consequences operator $T_{\mathbb{P}}$. In many fuzzy frameworks, the generalization of this operator is given through the supremum (of a set of values), which is associated with the existential quantifier. Hence, the presence of some errors in the data (noise) can disrupt the final result. This also happens in multi-adjoint logic programming [12, 13], which is a flexible fuzzy framework embedding the monotonic and residuated logic programing, and fuzzy logic programming frameworks [8, 19], among others. The basic

Partially supported by the 2014–2020 ERDF Operational Programme in collaboration with the State Research Agency (AEI) in projects TIN2016-76653-P and PID2019-108991GB-I00, and with the Department of Economy, Knowledge, Business and University of the Regional Government of Andalusia in project FEDER-UCA18-108612, and by the European Cooperation in Science & Technology (COST) Action CA17124.

J. Medina · J. A. Torné (✉)
Department of Mathematics, University of Cádiz, Cádiz, Spain
e-mail: joseantonio.torne@uca.es

J. Medina
e-mail: jesus.medina@uca.es

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

M. E. Cornejo et al. (eds.), *Computational Intelligence and Mathematics for Tackling Complex Problems 4*, Studies in Computational Intelligence 1040,
https://doi.org/10.1007/978-3-031-07707-4_3

operators in this framework are the adjoint pairs [4, 7], which generalize t-norms and its residuated operators, quantales, uninorms, etc. [5, 6].

The main goal of this paper is to provide a further generalization of the immediate consequences operator given in multi-adjoint logic programming [12] so that it is possible to reduce the existential feature of the supremum and so, relieve the possible presence of noise in the data. To proceed in this way, we will take advantage of the generalized quantifiers [2, 9, 14], due to the fact that generalized quantifiers provide a practical generalization of the existential quantifier, which is very versatile. These operators have already been used in diverse frameworks [1, 3] and in this paper we will use them for introducing the notion of quantified immediate consequences operator. Moreover, we will present two interesting examples, as well as the monotonicity property of this new operator, which is extremely required to ensuring that it is a right generalization of the classical one.

2 Preliminaries

In this section we are going to present some important results which will be useful in order to understand the proposed generalization of the $T_{\mathbb{P}}$. From now on, a graded set Ω, a set of propositional symbols Π, the Ω-algebra of well-formed formulas $\mathfrak{F}$ with carrier F_Ω, and a multi-adjoint Ω-algebra $\mathfrak{L} = \langle L, \leq, \&_1, \ldots, \&_n \rangle$ with carrier L will be fixed. More information about these notions is available for instance in [12].

Definition 1 A *multi-adjoint logic program* is a set $\mathbb{P}$ of weighted rules of the form $\langle (A \leftarrow_i \mathcal{B}) , \vartheta \rangle$ such that:

1. The rule $(A \leftarrow_i \mathcal{B})$ is a formula of $\mathfrak{F}$.
2. The confidence factor ϑ is an element (a truth-value) of L.
3. The head of the rule A is a propositional symbol of Π.
4. The body formula $\mathcal{B}$ is a formula of $\mathfrak{F}$ built from propositional symbols $B_1, \ldots, B_n$ $(n \geq 0)$ by the use of conjunctors $\&_1, \ldots, \&_n$ and $\wedge_1, \ldots, \wedge_k$, disjunctors $\vee_1, \ldots, \vee_l$ and aggregators $@_1, \ldots, @_m$.
5. Facts are rules with body $\top$.

Multi-adjoint logic program is the basic concept of the syntax of logic programming. On the other hand, semantics of a multi-adjoint logic program allows us to give meaning to formulas through its syntactic structure.

Definition 2 An *interpretation* is a mapping $I : \Pi \longrightarrow L$ which assigns a value of the complete lattice L to each propositional symbol. The set of all interpretations of the formulas defined by the Ω-algebra $\mathfrak{F}$ in the Ω-algebra $\mathfrak{L}$ is denoted as $\mathcal{I}_{\mathfrak{L}}$.

From homomorphic extension theorem, each interpretation uniquely extends to the set of formulas F_Ω, that is, $\hat{I} : F_\Omega \longrightarrow L$. The ordering $\leq$ defined in the set of truth-values L can be extended to the set of interpretations. This set $\mathcal{I}_{\mathfrak{L}}$ together with the point-wise ordering forms a complete lattice [12], denoted by $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$.

Definition 3 Given an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$ is *satisfied* by I if and only if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a *model* of $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.

It is possible to generalize the immediate consequences operator, given by van Emden and Kowalski [18], to the framework of multi-adjoint logic programs as follows [12].

Definition 4 The *immediate consequences operator* $T_{\mathbb{P}}^{\mathfrak{L}} \colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$, mapping interpretations to interpretations, is defined by considering

$$T_{\mathbb{P}}^{\mathfrak{L}}(I)(A) = \sup \left\{ \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid \langle (A \leftarrow_i \mathcal{B}), \vartheta \rangle \in \mathbb{P} \right\}$$

From now on, we will simply refer to it as $T_{\mathbb{P}}$. The following characterization of models of a multi-adjoint logic program $\mathbb{P}$ holds.

Theorem 1 ([12]) *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a model of $\mathbb{P}$ if and only if $T_{\mathbb{P}}(I) \sqsubseteq I$.*

In addition, the immediate consequences operator is monotonic in this framework, which is fundamental for the semantics.

Theorem 2 ([12]) *The operator $T_{\mathbb{P}}$ is monotonic.*

Hence, the semantics is given by the least fixed-point of the $T_{\mathbb{P}}$.

Theorem 3 ([12]) *For some ordinal α it is satisfied that $lfp(T_{\mathbb{P}}) = T_{\mathbb{P}}^{\alpha}(\triangle)$, where $\triangle$ is the least interpretation, that is, it is constantly the bottom element.*

Next, some basic aspects related to generalized quantifiers will be introduced, which will be fundamental for the new definition of $T_{\mathbb{P}}$ proposed in this paper.

Definition 5 Let U be a non-empty finite universe, $\mathcal{P}(U)$ the powerset of U and $\mu \colon \mathcal{P}(U) \longrightarrow [0, 1]$. We say that μ is a *fuzzy measure invariant with respect to the cardinality* if it holds:

1. $\mu(\varnothing) = 0$ and $\mu(U) = 1$.
2. For all $A, B \in \mathcal{P}(U)$ such that $A \subseteq B$, $\mu(A) \leq \mu(B)$.
3. If $|A| = |B|$, then $\mu(A) = \mu(B)$, for all $A, B \in \mathcal{P}(U)$, where $|\cdot|$ denotes the cardinality of a set.

From this definition, we can introduce the notion of generalized quantifier, where $\mathcal{F}(U) = \{C \mid C \colon U \to [0, 1]\}$ [3].

Definition 6 Let U be a non-empty finite universe, $\mathcal{P}(U)$ the powerset of U, $\mu \colon \mathcal{P}(U) \to [0, 1]$ a fuzzy measure invariant with respect to the cardinality and $\otimes$ an operator which is increasing on the first argument and it verifies the boundary conditions with respect to the top element. A mapping $Q_{\mu} \colon \mathcal{F}(U) \to [0, 1]$ defined, for all $C \in \mathcal{F}(U)$, as

$$Q_\mu(C) = \bigvee_{D \in \mathcal{P}(U)\setminus\{\varnothing\}} \left(\left(\bigwedge_{u \in D} C(u) \right) \otimes \mu(D) \right)$$

is called *generalized quantifier determined by the fuzzy measure* μ.

The following result provides a characterization of generalized quantifiers, which notably simplify its computation.

Theorem 4 ([3]) *Let Q_μ be a generalized quantifier in the non-empty finite universe $U = \{u_1, \ldots, u_n\}$, determined by a fuzzy measure invariant with respect to the cardinality μ. Then, for all $C \in \mathcal{F}(U)$,*

$$Q_\mu(C) = \bigvee_{i=1}^{n} C\left(u_{\pi(i)}\right) \otimes \mu\left(\{u_1, \ldots, u_i\}\right)$$

where π is a permutation on $\{1, 2, \ldots, n-1, n\}$ such that

$$C\left(u_{\pi(n)}\right) \leq C\left(u_{\pi(n-1)}\right) \leq \cdots \leq C\left(u_{\pi(2)}\right) \leq C\left(u_{\pi(1)}\right)$$

3 Fixed Point Semantics with Generalized Quantifiers

This section will introduce the aforementioned general definition of the immediate consequences operator based on generalized quantifiers. From now on, we will use a multi-adjoint logic program $\mathbb{P}$ over a set of propositional symbols Π and a multi-adjoint Ω-algebra $\mathfrak{L} = \langle [0, 1], \leq, \&_1, \ldots, \&_n \rangle$.

Definition 7 Let $\mathbb{P}$ be a non-empty multi-adjoint logic program, $\mu \colon \mathcal{P}(\mathbb{P}) \longrightarrow [0, 1]$ a fuzzy measure invariant with respect to the cardinality and $\otimes$ an operator which is increasing on the first argument and it verifies the boundary conditions with the top element. We define the *quantified immediate consequences operator* obtained from the generalized quantifier Q_μ as the mapping $T_{\mathbb{P}}^{Q_\mu} : \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$ given by

$$T_{\mathbb{P}}^{Q_\mu}(I)(A) = Q_\mu\left(C_I^A\right) = \bigvee_{D \in \mathcal{P}(\mathbb{P})} \left(\left(\bigwedge_{\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in D} C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta \rangle\right) \right) \otimes \mu(D) \right)$$

for all $I \in \mathcal{I}_{\mathfrak{L}}$ and $A \in \Pi$, where the mapping C_I^A is defined in Example 1.

First of all, we will see an example in which a comparison between this new definition of the $T_{\mathbb{P}}$ and the classical case is given.

Example 1 For all $A \in \Pi$ and $I \in \mathcal{I}_{\mathfrak{L}}$, it is satisfied that

$$T_{\mathbb{P}}^{Q^\exists}(I)(A) = Q^\exists\left(C_I^A\right) = T_{\mathbb{P}}(I)(A)$$

where $Q^{\exists}$ is the existential quantifier associated with the fuzzy measure $\mu^{\exists}$, recalled next, and $C_I^A : \mathbb{P} \to [0, 1]$ is the following mapping.

$$\mu^{\exists}(D) = \begin{cases} 0 \text{ if } D = \varnothing \\ \\ 1 \text{ if } D \neq \varnothing \end{cases} \qquad C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle\right) = \begin{cases} \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \text{ if } C = A \\ \\ 0 \qquad\quad \text{if } C \neq A \end{cases}$$

Considering the characterization of generalized quantifiers (Theorem 4), we have the following chain of equalities.

$$\begin{aligned} Q^{\exists}\left(C_I^A\right) &= \bigvee_{\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}} C_I^A\left(\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle\right) \\ &= \bigvee_{\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}} C_I^A\left(\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle\right) \\ &= \bigvee_{\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}} \left(\vartheta \dot{\&}_i \hat{I}(\mathcal{B})\right) \\ &= \sup\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid \langle A \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}\right\} \\ &= T_{\mathbb{P}}^{\mathfrak{L}}(I)(A) \end{aligned}$$

□

Next, we will take into account another example in which the generalized quantifier selects the second greatest element in the computation of $T_{\mathbb{P}}$ instead of the supremum.

Example 2 Consider a finite multi-adjoint logic program $\mathbb{P}$ with rules $\{u_1, \ldots, u_n\}$, an interpretation I, the mapping C_I^A given in Example 1 and the fuzzy measure $\mu^1 : \mathcal{P}(\mathbb{P}) \longrightarrow [0, 1]$ defined as

$$\mu^1(D) = \begin{cases} 0 & \text{if } D = \varnothing \text{ or } D = \{u_i\} \text{ for all } i \in \{1, \ldots, n\} \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

for all $D \in \mathcal{P}(\mathbb{P})$. We can order the elements $C_I^A(u_i)$ for each $i \in \{1, \ldots, n\}$ through the permutation π of the set of rules $\{u_1, \ldots, u_n\}$ satisfying that

$$C_I^A\left(u_{\pi(n)}\right) \leq C_I^A\left(u_{\pi(n-1)}\right) \leq \cdots \leq C_I^A\left(u_{\pi(2)}\right) \leq C_I^A\left(u_{\pi(1)}\right)$$

Taking into account this permutation, Definition 7 and the characterization of the generalized quantifiers given by Theorem 4, we obtain that

$$T_{\mathbb{P}}^{Q_{\mu^1}}(I)(A) = Q_{\mu^1}(C_I^A) = \bigvee_{i=1}^{n} C_I^A\left(u_{\pi(i)}\right) \otimes \mu^1\left(\{u_1, \ldots, u_i\}\right) = C_I^A\left(u_{\pi(2)}\right)$$

□

Furthermore, as this new operator is composed of monotonic mappings, it is also monotonic.

Theorem 5 *The immediate consequences operator* $T_{\mathbb{P}}^{\mathcal{Q}_\mu}$ *is increasing in* $\langle \mathcal{I}_{\mathfrak{L}}, \sqsubseteq \rangle$.

Thus, an interesting generalization of the fuzzy definition of immediate consequence operator has been introduced, which avoids the strong nature of the supremum. Other properties and the model semantics will be studied in a near future.

4 Conclusions and Future Work

In this article we have focused on the immediate consequences operator, which is fundamental in order to define the semantics of a multi-adjoint logic program. We have taken into account the problematic generated by the definition of the $T_{\mathbb{P}}$ through the supremum of a set of values, and we have given a generalization of this operator based on generalized quantifiers. Furthermore, a couple of interesting examples has been introduced and we have also shown that this new operator is monotonic, which is fundamental for given a fixed-point semantics in this framework. Hence, this will be the first goal as future work. Moreover, other research goals have been opened. For example, it is important to consider generalized quantifiers close to the supremum operator (existential quantifier) in order to reduce the strict feature of the supremum but preserving the original semantics in some sense. Hence, different measures will be studied in order to characterize these quantifiers. In addition, more flexible definitions of the immediate consequences operator will be studied, for instance, considering complete lattices instead of the unit interval or different quantifiers instead of only one.

References

1. Cao, N., Holčapek, M., Štěpnička, M.: Extensions of fuzzy relational compositions based on generalized quantifiers. Fuzzy Sets Syst. **339**, 73–98 (2018). Theme: Algebra and Set Theory
2. Cao, N., Stepnicka, M., Burda, M.: Fuzzy quantifiers and compositions of partial fuzzy relations employing dragonfly algebras. In: 2019 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2019, New Orleans, LA, USA, June 23–26, 2019, pp. 1–6. IEEE (2019)
3. Cornejo, M.E., Díaz-Moreno, J., López-Rodríguez, J., Medina, J.: Quantified concept-forming operators. In: CEUR Workshop Proceedings, vol. 2668, pp. 273–280 (2020)
4. Cornejo, M.E., Medina, J., Ramírez-Poussa, E.: A comparative study of adjoint triples. Fuzzy Sets Syst. **211**, 1–14 (2013)
5. Cornejo, M.E., Medina, J., Ramírez-Poussa, E.: Multi-adjoint algebras versus extended-order algebras. Appl. Math. & Inf. Sci. **9**(2L), 365–372 (2015)
6. Cornejo, M.E., Medina, J., Ramírez-Poussa, E.: Multi-adjoint algebras versus non-commutative residuated structures. Int. J. Approx. Reason. **66**, 119–138 (2015)

7. Cornejo, M.E., Medina, J., Ramírez-Poussa, E.: Algebraic structure and characterization of adjoint triples. Fuzzy Sets Syst. (2021)
8. Damásio, C.V., Pereira, L.M.: Monotonic and residuated logic programs. In: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01, pp. 748–759. Lecture Notes in Artificial Intelligence, vol. 2143 (2001)
9. Dvořák, A., Holčapek, M.: L-fuzzy quantifiers of type $\langle 1 \rangle$ determined by fuzzy measures. Fuzzy Sets Syst. **160**(23), 3425–3452 (2009). Theme: Computer Science
10. Julián-Iranzo, P., Rubio-Manzano, C.: A sound and complete semantics for a similarity-based logic programming language. Fuzzy Sets Syst. **317**, 1–26 (2017). Theme: Logic and Computer Science
11. Lloyd, J.: Foundations of Logic Programming. Springer, Berlin (1987)
12. Medina, J., Ojeda-Aciego, M., Vojtáš, P.: Multi-adjoint logic programming with continuous semantics. In: Logic Programming and Non-Monotonic Reasoning, LPNMR'01, pp. 351–364. Lecture Notes in Artificial Intelligence, vol. 2173 (2001)
13. Medina, J., Ojeda-Aciego, M., Vojtáš, P.: Similarity-based unification: a multi-adjoint approach. Fuzzy Sets Syst. **146**, 43–62 (2004)
14. Štěpnička, M., Holčapek, M.: Fuzzy relational compositions based on generalized quantifiers. In: Laurent, A., Strauss, O., Bouchon-Meunier, B., Yager, R.R. (eds.) Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp. 224–233. Springer International Publishing, Cham (2014)
15. Straccia, U., Ojeda-Aciego, M., Damásio, C.V.: On fixed-points of multivalued functions on complete lattices and their application to generalized logic programs. SIAM J. Comput. **38**(5), 1881–1911 (2009)
16. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific J. Math. **5**(2), 285–309 (1955)
17. van Emden, M.: Quantitative deduction and its fixpoint theory. J. Logic Program. **1**, 37–53 (1986)
18. van Emden, M., Kowalski, R.: The semantics of predicate logic as a programming language. J. ACM **23**(4), 733–742 (1976)
19. Vojtáš, P.: Fuzzy logic programming. Fuzzy Sets Syst. **124**(3), 361–370 (2001)

Anexo B

Preference in Multi-Adjoint Logic Programming based on Ordered Adjoint Pairs

Preference in Multi-adjoint Logic Programming Based on Ordered Adjoint Pairs

Jesús Medina and José Antonio Torné-Zambrano

1 Introduction

Preferences in decision-making have formally been treated from a mathematical point of view since the arrival of the notion of fuzzy set [11]. Thanks to membership functions (and taking into account the framework of decision-making problems based on the presence of uncertainty and lack of information), fuzzy sets have shown to be an adequate way to formalize the abstract notion of *preference*, as studied by Dubois and Prade in their papers about the three semantics of fuzzy sets [3–5]. All these papers based on decision making analysis has been studied together under the name of Fuzzy Decision Analysis (FDA).

The fuzzy logic programming framework considered in this paper will be the multi-adjoint one [1, 8], which models more complex situations than other logic programming frameworks. One of the main advantages of multi-adjoint logic programming is the possibility of taking into account diverse adjoint pairs in the design of a fuzzy logical program.

This paper complements the preliminary work given in [2] with the primary objective of analyzing the impact of the use of different implications in the rules of a multi-adjoint logic program, in order to study preferences in decision-making frameworks.

Partially supported by the 2014–2020 ERDF Operational Programme in collaboration with the State Research Agency (AEI) in projects PID2019-108991GB-I00 and PID2022-137620NB-I00, and with the Department of Economy, Knowledge, Business and University of the Regional Government of Andalusia in project FEDER-UCA18-108612, and by the European Cooperation in Science & Technology (COST) Action CA17124.

J. Medina · J. A. Torné-Zambrano (✉)
Department of Mathematics, University of Cádiz, Cádiz, Spain
e-mail: joseantonio.torne@uca.es

J. Medina
e-mail: jesus.medina@uca.es

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

M. E. Cornejo et al. (eds.), *Computational Intelligence and Mathematics for Tackling Complex Problems 5*, Studies in Computational Intelligence 1127,
https://doi.org/10.1007/978-3-031-46979-4_13

A study of how this choice affects the computation of the minimum models will be carried out, including different results. Moreover, some examples to illustrate the use of different (ordered) adjoint pairs will also be introduced.

2 Preliminaries

First of all, we recall some definitions useful in the understanding of the following sections. For more details see [8, 9].

Definition 1 Let $(P, \leq_P)$ and $\left(Q, \leq_Q\right)$ be two posets, A a set and the operators $\&\colon P \times A \to Q$ and $\swarrow\colon Q \times A \to P$. We say that $(\&, \swarrow)$ is a *right adjoint pair* w.r.t. P, A, Q if the following condition is verified

$$x \leq_P z \swarrow y \quad \text{iff} \quad x \& y \leq_Q z$$

for all $x \in P$, $y \in A$ and $z \in Q$. This pair will be simply called *adjoint pair* in the multi-adjoint logic programming framework.

The following example shows the main adjoint pairs we are going to consider in this study.

Example 1 The adjoint pairs $(\&_G, \leftarrow_G)$, $(\&_{\text{Ł}}, \leftarrow_{\text{Ł}})$ and $\left(\&_p, \leftarrow_p\right)$, where $\&_G$, $\&_{\text{Ł}}$ and $\&_p$ are the Gödel, Łukasiewicz and product adjoint conjunctor respectively, are defined, for all $x, y, z \in [0, 1]$, as

$$\&_G(x, y) = \min\{x, y\} \qquad z \leftarrow_G y = \begin{cases} 1 & \text{if} \quad y \leq z \\ z & \text{otherwise} \end{cases}$$

$$\&_{\text{Ł}}(x, y) = \max\{0, x + y - 1\} \qquad z \leftarrow_{\text{Ł}} y = \min\{1, 1 - y + z\}$$

$$\&_p(x, y) = x \cdot y \qquad z \leftarrow_p y = \begin{cases} 1 & \text{if} \quad y \leq z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$ □

Finally, the definition of the immediate consequence operator is recalled, which is given on a definite and propositional fuzzy logic program [8].

Definition 2 Let $\mathbb{P}$ be a multi-adjoint logic program. The *immediate consequence operator* $T_{\mathbb{P}}^{\mathfrak{L}}\colon \mathcal{I}_{\mathfrak{L}} \longrightarrow \mathcal{I}_{\mathfrak{L}}$, mapping interpretations to interpretations, is defined by considering

$$T_{\mathbb{P}}^{\mathfrak{L}}(I)(A) = \sup\left\{\vartheta \,\dot{\&}_i\, \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\right\}$$

3 Studying the Impact of Different Implications in Multi-adjoint Logic Programming

From now on, we will use the notation $\mathbb{P}_{ni}$ or $\mathbb{P}_{n\&_i}$ for a logic program in which every implication used in the construction of the n rules is the same, that is, $\leftarrow_i$, with their respective adjoint conjunctor $\&_i$. For example, $\mathbb{P}_{nG}$ is a multi-adjoint logic program with n rules, in which every rule is formed using the Gödel implication $\leftarrow_G$. Analogously, we could define logic programs $\mathbb{P}_{np}$ and $\mathbb{P}_{n\text{Ł}}$, with the product and Łukasiewicz conjunctors. We will use the notation $\mathbb{P}_{nij}$ or $\mathbb{P}_{n\&_i\&_j}$ for an arbitrary logic program considering the conjunctors $\&_i$ and $\&_j$. We will consider in this paper the linear complete lattice $([0, 1], \leq)$.

The first result is focused on the 'monotonicity' of the immediate consequence operator with respect to the considered adjoint pairs in the rules of the multi-adjoint program. Specifically, it shows that greater adjoint conjunctors provide greater immediate consequence operators.

Theorem 1 *Let* $\&_1 \colon [0, 1] \times [0, 1] \to [0, 1]$ *and* $\&_2 \colon [0, 1] \times [0, 1] \to [0, 1]$ *be two adjoint conjunctors verifying that* $\&_1 \leq \&_2$*, that is,* $x \&_1 y \leq x \&_2 y$*, for all* $(x, y) \in [0, 1] \times [0, 1]$*. Then, it is satisfied that* $T_{\mathbb{P}_{n\&_1}} \leq T_{\mathbb{P}_{n\&_2}}$.

As a consequence, we obtain the following chain for the usual Gödel, Łukasiewicz and product t-norms.

Corollary 1 *It is verified that* $T_{\mathbb{P}_{n\text{Ł}}} \leq T_{\mathbb{P}_{nP}} \leq T_{\mathbb{P}_{nG}}$.

In the following example we are going to study the effect on the value of the immediate consequence operator changing the implication used on the rule.

Example 2 Consider the program $\mathbb{P}_{2G\text{Ł}}$ formed with the rules $\langle A \leftarrow_G B_1, \vartheta_1 \rangle$ and $\langle A \leftarrow_{\text{Ł}} B_2, \vartheta_2 \rangle$ and the program $\mathbb{P}_{2p\text{Ł}}$ composed of the rules $\langle A \leftarrow_p B_1, \vartheta_1 \rangle$ and $\langle A \leftarrow_{\text{Ł}} B_2, \vartheta_2 \rangle$. The immediate consequence operator can be obtained, in both cases, as: $T_{\mathbb{P}_{2G\text{Ł}}}(I)(A) = \sup\left\{\vartheta_1 \dot{\&}_G \hat{I}(B_1), \vartheta_2 \dot{\&}_{\text{Ł}} \hat{I}(B_2)\right\}$ and $T_{\mathbb{P}_{2p\text{Ł}}}(I)(A) = \sup\left\{\vartheta_1 \dot{\&}_p \hat{I}(B_1), \vartheta_2 \dot{\&}_{\text{Ł}} \hat{I}(B_2)\right\}$. Therefore, the impact of the rule (relation among B_1 and A) in the modeled system is downplayed. However, depending on the interpretation I, it could not be effective as we show next. We can establish the following casuistry.

- If $\vartheta_2 \dot{\&}_{\text{Ł}} \hat{I}(B_2) \leq \vartheta_1 \dot{\&}_G \hat{I}(B_1)$, then it is verified that $\vartheta_1 \dot{\&}_p \hat{I}(B_1) \leq \vartheta_1 \dot{\&}_G \hat{I}(B_1)$ and, consequently $T_{\mathbb{P}_{2p\text{Ł}}} \leq T_{\mathbb{P}_{2G\text{Ł}}}$. Therefore, if a Gödel implication is replaced by a product one, then the change of the implication has a real impact in the computations obtaining lower values.
- If $\vartheta_1 \dot{\&}_G \hat{I}(B_1) \leq \vartheta_2 \dot{\&}_{\text{Ł}} \hat{I}(B_2)$, then $\vartheta_1 \dot{\&}_p \hat{I}(B_1) \leq \vartheta_2 \dot{\&}_{\text{Ł}} \hat{I}(B_2)$ and $T_{\mathbb{P}_{2G\text{Ł}}} = T_{\mathbb{P}_{2p\text{Ł}}}$. Consequently, the change of implication does not affect the value of the immediate consequence operator.
- If $\vartheta_2 \dot{\&}_{\text{Ł}} \hat{I}(B_2) \leq \vartheta_1 \dot{\&}_p \hat{I}(B_1)$, then it is verified that $\vartheta_1 \dot{\&}_{\text{Ł}} \hat{I}(B_1) \leq \vartheta_1 \dot{\&}_G \hat{I}(B_1)$ and $T_{\mathbb{P}_{2p\text{Ł}}} \leq T_{\mathbb{P}_{2G\text{Ł}}}$. The value of the immediate consequence operator would change

from $\vartheta_1 \dot{\&}_p \hat{I}(B_1)$ to $\vartheta_1 \dot{\&}_G \hat{I}(B_1)$. Therefore, the importance of the rule $\langle A \leftarrow B_1, \vartheta_1 \rangle$ is obvious.
– If $\vartheta_1 \dot{\&}_p \hat{I}(B_1) \leq \vartheta_1 \dot{\&}_{Ł} \hat{I}(B_2)$, then nothing can be deduced from this case. □

Theorem 2 *Consider two adjoint conjunctors* $\&_1 \colon [0,1] \times [0,1] \to [0,1]$ *and* $\&_2 \colon [0,1] \times [0,1] \to [0,1]$ *such that* $\&_1 \leq \&_2$. *Then, it is verified that* $T_{\mathbb{P}_{n\&_1}} \leq T_{\mathbb{P}_{n\&_1\&_2}} \leq T_{\mathbb{P}_{n\&_2}}$.

The following corollary shows some chains considering usual t-norms.

Corollary 2 *Consider a multi-adjoint logic program* $\mathbb{P}$ *and the family of adjoint conjunctors* $\Lambda = \{\&_{Ł}, \&_p, \&_G\}$, *together with their respective residuated implications. Then, it is verified that*

1. $T_{\mathbb{P}_{n\&_{Ł}}} \leq T_{\mathbb{P}_n} \leq T_{\mathbb{P}_{n\&_G}}$, *for all* $n \in \mathbb{N}$, *where* $T_{\mathbb{P}_n}$ *denotes a program with any possible combination of residuated implications of elements in* Λ.
2. $T_{\mathbb{P}_{n\&_p\&_{Ł}}} \leq T_{\mathbb{P}_{n\&_p}} \leq T_{\mathbb{P}_{n\&_p\&_G}}$.

Corollary 3 *Consider a multi-adjoint logic program* $\mathbb{P}$ *and the family of conjunctors* $T = \{\&_1, \&_2, \ldots, \&_t\}$, *with* $t \in \mathbb{N}$, $t \geq 2$, *verifying that* $\&_1 \leq \&_2 \leq \ldots \leq \&_t$. *Then, it is satisfied that*

$$T_{\mathbb{P}_{n\&_1}} \leq T_{\mathbb{P}_{n\&_1\&_2}} \leq T_{\mathbb{P}_{n\&_2}} \leq \ldots \leq T_{\mathbb{P}_{n\&_{t-1}}} \leq T_{\mathbb{P}_{n\&_{t-1}\&_t}} \leq T_{\mathbb{P}_{n\&_t}}$$

In the following example, we will apply these results to a particular case in which we use a logic program in order to distinguish between covid and cold.

Example 3 Consider the multi-adjoint logic program $\mathbb{P}$ composed of the following rules.

$$\begin{aligned}
&\texttt{covid} \overset{0.5}{\leftarrow} \texttt{diarrhoea} \wedge_{Ł} \texttt{headache}\\
&\texttt{covid} \overset{0.9}{\leftarrow} \texttt{moderate fever}\\
&\texttt{covid} \overset{0.8}{\leftarrow} \texttt{loss of smell and taste}\\
&\texttt{covid} \overset{0.4}{\leftarrow} \texttt{sore throat} \wedge_{Ł} \texttt{muscle pain}\\
&\texttt{covid} \overset{0.7}{\leftarrow} \texttt{cough}\\
&\texttt{covid} \overset{0.6}{\leftarrow} \texttt{stuffy and runny nose}\\
&\qquad\qquad \wedge_{Ł} \texttt{shortness of breath}\\
&\texttt{covid} \overset{0.6}{\leftarrow} \texttt{fatigue}
\end{aligned}
\qquad
\begin{aligned}
&\texttt{cold} \overset{0.6}{\leftarrow} \texttt{diarrhoea} \wedge_{Ł} \texttt{moderate fever}\\
&\texttt{cold} \overset{0.1}{\leftarrow} \texttt{shortness of breath}\\
&\texttt{cold} \overset{0.8}{\leftarrow} \texttt{fatigue} \wedge_{Ł} \texttt{muscle pain}\\
&\texttt{cold} \overset{0.5}{\leftarrow} \texttt{loss of smell and taste}\\
&\qquad\qquad \wedge_{Ł} \texttt{sore throat}\\
&\texttt{cold} \overset{0.6}{\leftarrow} \texttt{stuffy and runny nose}\\
&\texttt{cold} \overset{0.9}{\leftarrow} \texttt{cough}\\
&\texttt{cold} \overset{0.7}{\leftarrow} \texttt{headache}
\end{aligned}$$

These rules are completed with observations about each particular person, in the form of *facts*. For example, one specific individual can show the following symptoms.

$$\begin{aligned}
\texttt{moderate fever} &\overset{0.6}{\leftarrow} \top\\
\texttt{shortness of breath} &\overset{0.1}{\leftarrow} \top\\
\texttt{loss of smell and taste} &\overset{0.1}{\leftarrow} \top\\
\texttt{stuffy and runny nose} &\overset{0.7}{\leftarrow} \top
\end{aligned}
\qquad
\begin{aligned}
\texttt{sore throat} &\overset{0.5}{\leftarrow} \top\\
\texttt{cough} &\overset{0.8}{\leftarrow} \top\\
\texttt{fatigue} &\overset{0.4}{\leftarrow} \top
\end{aligned}
\qquad
\begin{aligned}
\texttt{diarrhoea} &\overset{0.6}{\leftarrow} \top\\
\texttt{headache} &\overset{0.7}{\leftarrow} \top\\
\texttt{muscle pain} &\overset{0.2}{\leftarrow} \top
\end{aligned}$$

Taking into account the current health crisis and the great ease with which a person can get covid-19, it is advisable to do the diagnosis in secure areas, without risking of going to a clinic. With this purpose, multi-adjoint logic programming can be used in order to suggest a first diagnosis of the possible disease of the user. In this case, the introduced program will help to identify the possible disease of the patient, distinguishing between covid-19 and cold. We must bear in mind that this is a simple example with which to illustrate in the most didactic way the use of preferences in multi-adjoint logic programming. By Corollary 2, we can associate different degrees of preference with the conjunctors $\&_G$, $\&_Ł$ and $\&_P$.

- $\&_Ł$ is a low degree of preference. In a decision-making problem, $\&_Ł$ will be associated with the idea of "*I prefer not*".
- $\&_P$ is a medium degree of preference, corresponding with the idea of "*I neither prefer it nor stop preferring it*".
- $\&_G$ is a high degree of preference. In a decision-making problem, $\&_G$ will be associated with the idea of "*I prefer*".

Now, we are going to understand this problem in different contexts.

• *First situation*. We can assume that the user shows the symptoms aforementioned, but this person has barely left his house in the last two weeks, because he works on-line. He has only gone for shopping and walking. In addition, he met with a friend one week ago and his wife, who is a teacher, strictly complies with all security measures against covid-19. Therefore, at first, it is more probable that this person has a cold than covid-19, but we cannot discard this last possibility. Consequently, taking into account that his "degree of preference" of cold is higher, then in this situation we are going to use Gödel conjunctors for the rules with head `cold`, while Łukasiewicz one for the rules with head `covid`. Next, we calculate the value of the immediate consequence operator for the propositional symbol `covid`. We recall how the least fixed point of the immediate consequence operator is calculated starting from the least interpretation. The first is trivially: $T_{\mathbb{P}_Ł}(\triangle)(\texttt{covid}) = 0$.

$$
\begin{aligned}
T^2_{\mathbb{P}_Ł}(\triangle)(\texttt{covid}) &= T_{\mathbb{P}_Ł}\left(T_{\mathbb{P}_Ł}(\triangle)(\texttt{covid})\right)\\
&= \sup\Big\{0.5\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{diarrhoea} \wedge_Ł \texttt{headache}),\\
&\qquad 0.9\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{moderate fever}), 0.7\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{cough}),\\
&\qquad 0.8\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{loss of smell and taste}),\\
&\qquad 0.4\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{sore throat} \wedge_Ł \texttt{muscle pain}),\\
&\qquad 0.6\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{stuffy and runny nose}\\
&\qquad\qquad \wedge_Ł \texttt{shortness of breath}),\\
&\qquad 0.6\dot{\&}_Ł \widehat{T_{\mathbb{P}_Ł}(\triangle)}(\texttt{fatigue})\Big\}\\
&= \sup\big\{0.5\dot{\&}_Ł (0.6 \wedge_Ł 0.7), 0.9\dot{\&}_Ł 0.6, 0.7\dot{\&}_Ł 0.8, 0.4\dot{\&}_Ł (0.5 \wedge_Ł 0.2),\\
&\qquad 0.8\dot{\&}_Ł 0.1, 0.6\dot{\&}_Ł (0.7 \wedge_Ł 0.1), 0.6\dot{\&}_Ł 0.4\big\}\\
&= \sup\{0, 0.5, 0.5, 0, 0, 0, 0\} = 0.5
\end{aligned}
$$

Analogously, it can be shown that $T_{\mathbb{P}^3_{Ł}}(\triangle)(\texttt{covid}) = 0.5$, $T_{\mathbb{P}_G}(\triangle)(\texttt{covid}) = 0$, $T_{\mathbb{P}^2_G}(\triangle)(\texttt{covid}) = 0.8$ and $T_{\mathbb{P}^3_G}(\triangle)(\texttt{covid}) = 0.8$. We can see that, in this situation, it is more probable that the person has cold than covid-19 because the value of the immediate consequence operator of `cold` (with the Gödel implication) is much higher than the value associated with `covid`.

• *Second situation*. We are going to proceed in reverse of the previous case. We can assume that the person was in contact with another person, without using the masks and without keeping the safety distance. Given these circumstances, it could be that the person thought that he could have been infected because of his friend, so the "preference degree" is now going to be covid-19, although it cannot be ruled out that it could also be the cold. Therefore, in the rules with head `covid`, we will use Gödel implications, while in the rules with head `cold`, Łukasiewicz ones. In this case, we can check that the value obtained is the same. Therefore, although the possibility that this person may present coronavirus rises (considerably when compared to using the other conjunctors), it is not enough to overcome the possibility of cold.

We obtain that $T_{\mathbb{P}_G}(\triangle)(\texttt{covid}) = 0$, $T_{\mathbb{P}^2_G}(\triangle)(\texttt{covid}) = 0.7$, $T_{\mathbb{P}^3_G}(\triangle)(\texttt{covid}) = 0.7$, $T_{\mathbb{P}_{Ł}}(\triangle)(\texttt{cold}) = 0$, $T_{\mathbb{P}^2_{Ł}}(\triangle)(\texttt{cold}) = 0.7$ and $T_{\mathbb{P}^3_{Ł}}(\triangle)(\texttt{cold}) = 0.7$.

• *Third situation*. Finally, if we are not sure about one of the symptoms, for example, rules with `fatigue` and `muscle pain`, then we can use in those rules a Łukasiewicz conjunctor, while in the rest of the rules we can consider a product or Gödel one. □

4 Conclusions and Future Work

The impact of the use of different ordered adjoint pairs has been introduced in this paper. This preliminary study has shown this ordering is naturally translated into the computation of the minimal model. Furthermore, an illustrative example has been presented containing some ideas about how the selections of adjoint pairs, depending on the preference of the user, can be done. In the future, more properties and interpretations of the selections will be studied. For instance, the relationship with the theory introduced in [6, 7, 10] will be analyzed.

References

1. Cornejo, M.E., Medina, J., Ramírez-Poussa, E.: Multi-adjoint algebras versus non-commutative residuated structures. Int. J. Approx. Reason. **66**, 119–138 (2015)
2. Cornejo, M.E., Medina, J., Ramírez-Poussa, E., Rubio-Manzano, C.: Multi-adjoint concept lattices, preferences and Bousi prolog. In: Flores, V., Gomide, F., Janusz, A., Meneses, C., Miao, D., Peters, G., Ślęzak, D., Wang, G., Weber, R., Yao, Y. (eds.) Rough Sets, pp. 331–341. Springer International Publishing, Cham (2016)

3. Dubois, D.: The role of fuzzy sets in decision sciences: old techniques and new directions. Fuzzy Sets Syst. **184**(1), 3–28 (2011). Preference Modelling and Decision Analysis (Selected Papers from EUROFUSE 2009)
4. Dubois, D., Prade, H.: The three semantics of fuzzy sets. Fuzzy Sets Syst. **90**(2), 141–150 (1997). Fuzzy Sets: Where Do We Stand? Where Do We Go?
5. Dubois, D., Prade, H.: Gradualness, uncertainty and bipolarity: making sense of fuzzy sets. Fuzzy sets Syst. **192**, 3–24 (2012)
6. Eiter, T., Fink, M.: Uniform Equivalence of Logic Programs Under the Stable Model Semantics. vol. 2916, pp. 224–238 (2003)
7. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Trans. Comput. Log. **2**, 526–541 (2001)
8. Medina, J., Ojeda-Aciego, M., Vojtáš, P.: Multi-adjoint logic programming with continuous semantics. In: Logic Programming and Non-monotonic Reasoning, LPNMR'01. Lecture Notes in Artificial Intelligence, vol. 2173, pp. 351–364 (2001)
9. Medina, J., Ojeda-Aciego, M., Vojtáš, P.: Similarity-based unification: a multi-adjoint approach. Fuzzy Sets Syst. **146**, 43–62 (2004)
10. Woltran, S.: Characterizations for Relativized Notions of Equivalence in Answer Set Programming, vol. 3229, pp. 161–173 (2004)
11. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)

Anexo C

Hypergraph logic program representation versus stratified programs

Hypergraph logic program representation versus stratified programs*

David Lobo[1][0000−0002−7678−2331], Jesús Medina[1][0000−0002−3931−5873], José R. Portillo[2,3][0000−0002−0940−068X], and José A. Torné-Zambrano[1][0000−0002−9897−916X]

[1] Department of Mathematics. University of Cádiz, Spain
{david.lobo,jesus.medina,joseantonio.torne}@uca.es
https://www.uca.es/mcis
[2] Department of Applied Mathematics 1, University of Sevilla, Spain
https://ma1.us.es
[3] University Institute for Research in Mathematics (IMUS), Seville, Spain.
https://imus.us.es
josera@us.es

Abstract. Multi-adjoint normal logic programming is a general non-monotonic logic programming framework, which makes it ideal for modeling complex scenarios. Hypergraph representation has been proved to be an appropriate tool in the study of different properties of a logic program, whereas the use of a stratification has provided interesting results related to the existence and unicity of stable models. In this paper, we will see the relation between the p-condensation graph of a program and its "optimal" stratification.

Keywords: Logic Programming · Hypergraphs · Negation operator.

1 Introduction

Logic programming arises as a mathematical tool for expressing and solving problems through declarative specifications, becoming particularly powerful when dealing with knowledge-based systems and automated reasoning [14, 18]. Among all the generalizations of logic programming, it stands out the multi-adjoint normal one (MANLP) [4, 5], allowing to use different adjoint pair in the construction of the rules of the programs [6], together with less restrictive properties in the considered connectives and the use of negation operators, providing a more flexible framework.

Taking into account that one of the main demanding tasks in MANLP is the study of conditions to ensure the existence and the unicity of stable models [11,

* Partially supported by the 2014–2020 ERDF Operational Programme in collaboration with the State Research Agency (AEI) in projects PID2019-108991GB-I00 and PID2022-137620NB-I00, and the Ecological and Digital Transition Projects 2021 of the Ministry of Science and Innovation in project TED2021-129748B-I00.

15], we can understand the importance of stratifications in logic programming. Indeed, the determination of stratified logic programs is a relevant sufficient condition in order to determine whether a logic program has a (unique) stable model [1, 17].

Furthermore, multi-adjoint logic programs have been represented by hypergraphs obtaining interesting results. For example, this relationship was used in [8, 9] to the termination property of the immediate consequences operator. This representation was extended in [10].

In this paper, we will provide a hypergraph representation of a MANLP completely characterizing the considered program, complementing the one given in [9, 10]. Moreover, we will show that from the p-condensation graph of the program $\mathbb{P}$, we can define an "optimal" stratification of the given program. As a consequence, it also makes possible to compute the least model of the subprogram associated with each stratum in parallel.

2 Multi-adjoint normal logic programming

The algebraic structure considered in this framework is a local multi-adjoint normal Σ-algebra, which considers a bounded lattice endowed with a negation operator and adjoint pairs, which generalize left-continuous t-norms and their residuated implications. For more details, the reader can see [4, 16].

In this paper, we will consider a Σ-algebra $\mathfrak{L}$ based on a multi-adjoint lattice [7] $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$, a negation operator $\neg$, and other operators, such as conjunctors $\wedge_1, \ldots, \wedge_k$, disjunctors $\vee_1, \ldots, \vee_l$ and general aggregators $@_1, \ldots, @_h$. Considering the multi-adjoint Σ-algebra $\mathfrak{L}$, a countable set of propositional symbols Π and the algebra of well-formed formulas $\mathfrak{F}$, we introduce the definition of multi-adjoint normal logic program [4, 16].

Definition 1. *A* multi-adjoint normal logic program*, MANLP in short, is a set of rules of the form* $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$ *such that:*

1. *The* rule $A \leftarrow_i \mathcal{B}$ *is a formula of* $\mathfrak{F}$.
2. *The* confidence factor ϑ *is an element (a truth-value) of* L.
3. *The* head *of the rule* A *is a propositional symbol of* Π.
4. *The* body *formula* $\mathcal{B}$ *is a formula of* $\mathfrak{F}$ *of the form* $@[B_1, \ldots, B_s, \neg B_{s+1}, \ldots, \neg B_r]$ *built from propositional symbols* $B_1, \ldots, B_r$ *(*$r \geq 0$*,* $B_i \neq B_j$*, for* $i \neq j$ *) by the use of conjunctors* $\&_1, \ldots, \&_n$ *and* $\wedge_1, \ldots, \wedge_k$*, disjunctors* $\vee_1, \ldots, \vee_l$*, aggregators* $@_1, \ldots, @_m$ *and elements of* L *(which composition is represented by* $@$*).*
5. Facts *are rules with the body* $\top$.

Example 1. In this example in the unit interval $L = [0, 1]$, the adjoint pairs corresponding to the product, Gödel and Lukasiewicz t-norms, $(\&_{\mathrm{P}}, \leftarrow_{\mathrm{P}}), (\&_{\mathrm{G}}, \leftarrow_{\mathrm{G}})$, $(\&_{\mathrm{Ł}}, \leftarrow_{\mathrm{Ł}})$ are considered, together with the weighted sums $@_{(2,1)}$ and $@_{(1,3)}$ defined as $@_{(2,1)}(x, y) = \dfrac{2x + y}{3}$ and $@_{(1,3)}(x, y) = \dfrac{x + 3y}{4}$, for every $(x, y) \in [0, 1]^2$. Moreover, the negation $\neg$, defined as $\neg(x) = 1 - x$ for $x \in [0, 1]$, will also be

taken into account in the program. Specifically, the following normal program $\mathbb{P}$ will be analyzed in the rest of the paper:

$$\begin{array}{lll}
\langle \mathrm{c} \leftarrow_{\mathrm{P}} \mathrm{n} \;\&_{\mathrm{P}}\; \neg \mathrm{u}, 0.8\rangle & \langle \mathrm{p} \leftarrow_{\mathrm{P}} \mathrm{n}, 0.9\rangle & \langle \mathrm{f} \leftarrow_{\mathrm{P}} 1.0, 0.1\rangle \\
\langle \mathrm{n} \leftarrow_{\mathrm{P}} \mathrm{c}, 0.8\rangle & \langle \mathrm{m} \leftarrow_{\mathrm{P}} \mathrm{n} \;\&_{\mathrm{G}}\; \mathrm{a}, 0.8\rangle & \langle \mathrm{n} \leftarrow_{\mathrm{P}} 1.0, 0.5\rangle \\
\langle \mathrm{n} \leftarrow_{\mathrm{P}} @_{(2,1)}(\neg \mathrm{f}, \neg \mathrm{h}), 0.6\rangle & \langle \mathrm{f} \leftarrow_{\mathrm{P}} \mathrm{u}, 0.9\rangle & \langle \mathrm{u} \leftarrow_{\mathrm{P}} 1.0, 0.2\rangle \\
\langle \mathrm{h} \leftarrow_{\mathrm{P}} \mathrm{f}, 0.7\rangle & \langle \mathrm{a} \leftarrow_{\mathrm{P}} @_{(1,3)}(\neg \mathrm{f}, \neg \mathrm{u}), 1.0\rangle & \langle \mathrm{u} \leftarrow_{\mathrm{P}} 1.0, 0.2\rangle \\
\langle \mathrm{u} \leftarrow_{\mathrm{G}} \mathrm{h} \,\&_{\mathrm{Ł}}\, \mathrm{f}, 0.7\rangle &
\end{array}$$

Note that, in non-commutative aggregators such as $@_{(1,3)}$ or $@_{(2,1)}$, the order of variables in the input follows the enumeration defined by the set Π. In our example, we must write $@_{(1,3)}(\neg \mathrm{f}, \neg \mathrm{u})$ instead of $@_{(3,1)}(\neg \mathrm{u}, \neg \mathrm{f})$.

The semantics is given by the notion of model:

Definition 2. *Given a mapping (called interpretation) $I\colon \Pi \to L$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ is* satisfied *by I, if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$, where $\hat{I}$ is the unique homomorphic extension of I on $\mathfrak{F}$. An interpretation $I\colon \Pi \to L$ is a* model *of a multi-adjoint normal logic program $\mathbb{P}$ if all weighted rules in $\mathbb{P}$ are satisfied by I.*

Now, we recall the notion of stratified logic program, according to the seminal version of the notion, simultaneously introduced in [1, 17]. Given the set of propositional symbols appearing in the program $\mathbb{P}$, which will be denoted as $\Pi_{\mathbb{P}}$, a *stratification of* $\mathbb{P}$ is a mapping $||\cdot||\colon \Pi_{\mathbb{P}} \to \mathbb{Z}^+$ such that, for each rule of $\mathbb{P}$ of the form $\langle p \leftarrow_i @\,[p_1, \ldots, p_m, \neg p_{m+1}, \ldots, \neg p_n]\,; \vartheta\rangle$, the following statements hold:

- $||p_i|| \leq ||p||$ for each $i \in \{1, \ldots, m\}$.
- $||p_i|| < ||p||$ for each $i \in \{m+1, \ldots, n\}$.

Given a propositional symbol $p \in \Pi_{\mathbb{P}}$, the *stratum* of p is the integer $||p||$. The existence of stratifications characterizes the so-called stratified multi-adjoint normal logic programs. A multi-adjoint normal logic program $\mathbb{P}$ is *stratified* if there exists a stratification of $\mathbb{P}$. Indeed, if $||\cdot||$ is a stratification of $\mathbb{P}$, then the unique stable model of $\mathbb{P}$ can be computed in terms of a certain sequence of interpretations $\{M_1^{||\cdot||}, M_2^{||\cdot||}, \ldots\}$ related to the strata of $\mathbb{P}$ with respect to $||\cdot||$.

3 Normal logic programs through hypergraphs

Basic notions of (hyper)graph theory can be found in [2]. A *directed hypergraph* is a pair (V, E), where elements of E are *directed hyperedges* or *hyperarcs*, represented as ordered pairs $e = (T(e), H(e))$, with disjoint subsets of vertices $T(e)$ (the *tail*) and $H(e)$ (the *head*) [12]. Hereafter, directed hypergraphs are simply referred to as hypergraphs. An *edge labeling* is a function from E to labels, and a hypergraph with an edge labeling is an *edge-labeled hypergraph*. A *B-graph* is a hypergraph where all hyperarcs have only one element in their heads [12]. This paper focuses on labeled B-graphs, a natural representation of rules in a

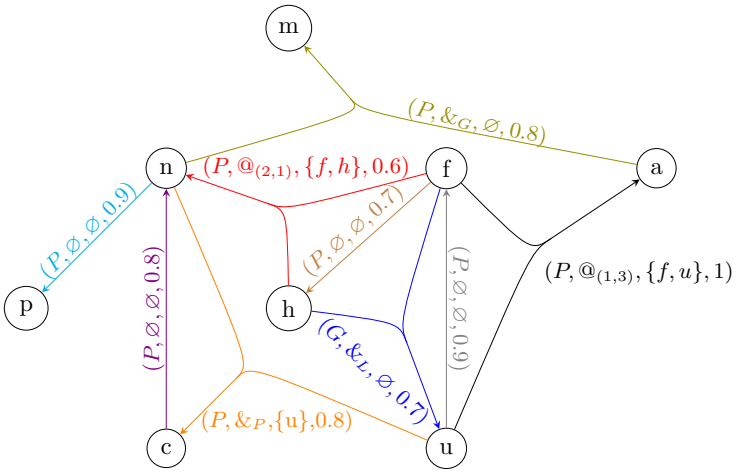

Fig. 1. Edge-labeled B-graph associated with the program given in Example 1. The facts of the program (rules 8, 9 and 10 of Example 1) are not shown by simplicity.

multi-adjoint normal logic program. This section will illustrate how a flexible MANLP can be represented by a specific edge-labeled directed B-graph.

Given a multi-adjoint normal logic program $\mathbb{P}$, we compute an associated B-graph $\mathcal{H}_{\mathbb{P}}$. Vertices correspond to propositional symbols in $\Pi_{\mathbb{P}}$, and hyperarcs are assigned for each rule. Labels include a 4-vector $(i, @, \{B_{s+1}, \ldots, B_r\}, \vartheta)$, with special cases for single symbols in the body. Figure 1 shows the B-graph associated with Example 1. The resulting hypergraph is an edge-labeled directed B-graph, and the original program can be reconstructed from it [9, 10]. This characterization allows leveraging existing results and algorithms for directed hypergraphs in the analysis of fuzzy logic normal programs.

Given a directed hypergraph $\mathcal{H} = (V, E)$, two vertices $u, v \in V$, and a natural number q, a *path* in $\mathcal{H}$ from u to v of length q is a sequence of nodes and hyperarcs: $P_{uv} = \langle v_1, E_1, v_2, E_2, \ldots, v_q, E_q, v_{q+1} \rangle$ where: $v_1 = u$, $v_{q+1} = v$, $u \in T(E_1), v \in H(E_q)$, and $v_i \in H(E_{i-1}) \cap T(E_i)$, for all $i \in \{2, ..., q\}$. The trivial sequence $P_{vv} = \langle v \rangle$, with $v \in V$, will be a path of length 0. Given a hypergraph $\mathcal{H}$, vertices u and v are *strongly path-connected* if there is a path from u to v and vice versa. A *strongly path-connected component* (SPC-component) is an equivalence class under this relation.

As a consequence, vertices in each cycle in a hypergraph belong to the same SPC-component. Example 1 has three SPC-components: {f, h, u}, {c, n}, and {a}.

SPC-components provide an interesting partition of the hyperarcs into two subsets. A *d-hyperarc* has vertices in **d**ifferent SPC-components in its tail and head; an *s-hyperarc* has at least one vertex in the **s**ame SPC-component. Figure 2 depicts the labeled B-graph associated with Example 1, highlighting SPC-

components and d-/s-hyperarcs. Labels are also included. Notice that, the negation operators only appear in the d-hyperarcs.

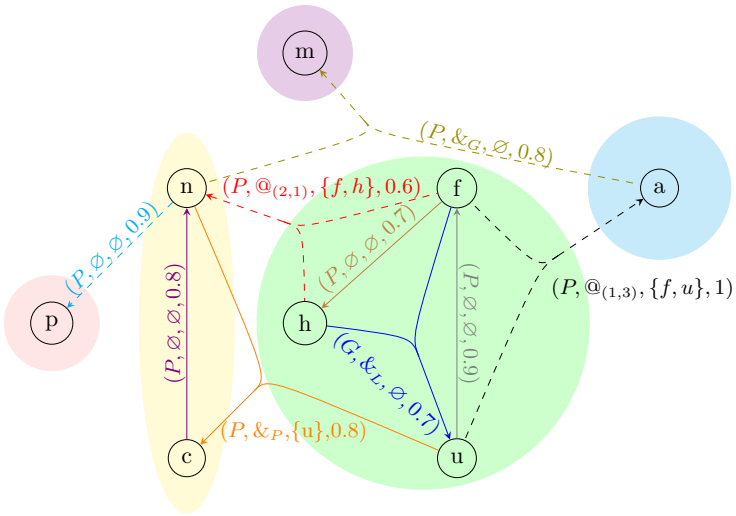

Fig. 2. SPC-components of the edge-labeled B-graph associated with the program given in Example 1. The d-hyperarcs are shown with dashed lines.

The *path-condensation* (*p-condensation*, for short) of a hypergraph $\mathcal{H}$, denoted as $C(\mathcal{H})$, is a digraph whose vertices are SPC-components, with arcs connecting components based on hyperarc relationships [13]. Figure 3 shows the p-condensation of the hypergraph in Figure 1.

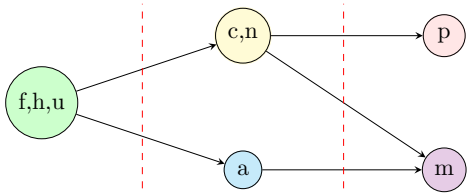

Fig. 3. p-condensation digraph of the hypergraph in Figure 1. Vertices are labelled with the vertices of the corresponding SPC of the hypergraph.

4 Application of the hypergraph representation

This section shows that the hypergraph representation provides a mechanism for detecting whether a program $\mathbb{P}$ is stratified and, in this case, providing an "optimal" stratification.

As aforementioned, stratifications characterize the unique stable model of a stratified MANLP $\mathbb{P}$. However, since many stratifications can be defined from a program, even if we limit to non-isomorphic stratifications, then there might be multiple forms for computing the (unique) stable model of $\mathbb{P}$. For example, a stratification $||\cdot||$ with many strata requires many elements of the sequence $\{M_i^{||\cdot||}\}$, i.e. many steps, to obtain the unique stable model, thus slowing down its computation.

Furthermore, and more important, the constructive method of the stable model implies the computation of a succession of interpretations in a sequential way, avoiding parallel computations. The hypergraph representation of MANLP given in the previous section provides an interesting tool to deal with the two previous issues, and to optimize the computation of the unique stable model of stratified MANLP. Specifically, we obtain the following result.

Theorem 1. *Given a MANLP $\mathbb{P}$. If the labels of s-hyperarcs of $\mathcal{H}_\mathbb{P}$ do not contain negative variables in the same* SPC*-component of the s-hyperarc head, then the p-condensation graph $\mathcal{C}(\mathcal{H}_\mathbb{P})$ provides a stratification with the least possible number of non-empty strata, satisfying that $\|v_i\| < \|v_j\|$ for every couple of vertices v_i, v_j of $\mathcal{C}(\mathcal{H}_\mathbb{P})$ with a directed path from v_i to v_j.*

For example, in the previous particular program, we can see in Fig. 3 that the p-condensation graph offers an stratification $\|\cdot\|$ with three strata: $\|\mathrm{f}\| = \|\mathrm{h}\| = \|\mathrm{u}\| = 0$, $\|\mathrm{c}\| = \|\mathrm{n}\| = \|\mathrm{a}\| = 1$, and $\|\mathrm{p}\| = \|\mathrm{m}\| = 2$. Clearly, other stratifications can be defined on $\mathbb{P}$ satisfying that $\|v_i\| < \|v_j\|$ for every v_i, v_j of $\mathcal{C}(\mathcal{H}_\mathbb{P})$ with a directed path from v_i to v_j, but all of them have three or more strata.

Therefore, the stratification provided by the hypergraph representation of a MANLP $\mathbb{P}$ optimizes the number of strata. Furthermore, it also allows the computation of the least model of the subprogram associated with every stratum in parallel, with respect to each independent SPC-component of the stratum.

5 Conclusions and future work

We have complemented previous advances in order to completely characterize a MANLP through a hypergraph. We have also shown that, if a given MANLP is stratified, then the p-condensation graph of the hypergraph associated with the program provides a stratification with the least number of non-empty strata, satisfying that if two vertices of the graph are connected they must be in different strata, also optimizing parallel computations.

In the future, more properties and advantages will be obtained from the relationship between hypergraph theory and logic programming. For example, we will study the theoretical complexity of the computation of the stratification

given by Theorem 1 and it will be compared with other traditional methods to compute stable models. Furthermore, we will apply the hypergraph representation to obtain new results in the computation of the semantics of modular logic programs [3].

References

1. K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1988.
2. C. Berge. *Graphs and Hypergraphs*. Elsevier Science Ltd, 1985.
3. P. Cabalar, J. Fandinno, and Y. Lierler. Modular answer set programming as a formal specification language. *Theory and Practice of Logic Programming*, 20(5):767–782, 2020.
4. M. E. Cornejo, D. Lobo, and J. Medina. Syntax and semantics of multi-adjoint normal logic programming. *Fuzzy Sets and Systems*, 345:41 – 62, 2018.
5. M. E. Cornejo, D. Lobo, and J. Medina. Extended multi-adjoint logic programming. *Fuzzy Sets and Systems*, 388:124–145, 2020. Logic.
6. M. E. Cornejo and J. Medina. Impact Zadeh's theory to algebraic structures. multi-adjoint algebras. *Journal of Pure and Applied Mathematics*, 12:126–141, 2021.
7. M. E. Cornejo, J. Medina, and E. Ramírez-Poussa. Multi-adjoint algebras versus non-commutative residuated structures. *International Journal of Approximate Reasoning*, 66:119–138, 2015.
8. J. C. Díaz-Moreno, J. Medina, and J. R. Portillo. Towards the use of hypergraphs in multi-adjoint logic programming. *Studies in Comp. Intelligence*, 796:53–59, 2019.
9. J. C. Díaz-Moreno, J. Medina, and J. R. Portillo. Fuzzy logic programs as hypergraphs. termination results. *Fuzzy Sets and Systems*, 445:22–42, 2022.
10. J. C. Díaz-Moreno, J. Medina, and J. R. Portillo. Hypergraphs in logic programming. In Z. Bouraoui and S. Vesic, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 442–452, Cham, 2024. Springer.
11. M. Fitting. The family of stable models. *The Journal of Logic Programming*, pages 17(2–4):197–225, 1993.
12. G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Appl. Math.*, 42(2-3):177–201, apr 1993.
13. F. Harary, R. Z. Norman, and D. Cartwright. *Structural Models: An Introduction to the Theory of Directed Graphs*. John Wiley & Sons, New York, 4 edition, 1965.
14. J. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.
15. N. Madrid and M. Ojeda-Aciego. On the existence and unicity of stable models in normal residuated logic programs. *International Journal of Computer Mathematics*, 89(3):310–324, 2012.
16. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. *Lecture Notes in Artificial Intelligence*, 2173:351–364, 2001.
17. A. Van Gelder. Negation as failure using tight derivations for general logic programs. *The Journal of Logic Programming*, 6(1-2):109–133, 1989.
18. C. Zhang, L. Chen, Y.-P. Zhao, Y. Wang, and C. L. P. Chen. Graph enhanced fuzzy clustering for categorical data using a bayesian dissimilarity measure. *IEEE Transactions on Fuzzy Systems*, 31(3):810–824, 2023.

Anexo D

Revisiting the immediate consequences operator on first-order logic

Revisiting immediate consequences operator on first-order logic programming*

Jesús Medina and José Antonio Torné-Zambrano

Department of Mathematics. University of Cádiz. Spain
{jesus.medina, joseantonio.torne}@uca.es

Abstract. Logic programming is a fundamental paradigm based on formal logic with relevant features in the design of automatic intelligence systems. The computation of the consequences from a given dataset, modeled by logic rules, is given by the immediate consequences operator. This paper clarifies the definition of this operator in the first-order logic programming framework and remarks the consideration of two sets in the computation of the immediate consequences operator.

1 Introduction

Among all the generalizations of logic programming, it stands out the residuated and multi-adjoint approaches [4,14], which are able to capture different extensions developed for different purposes such as Generalized Annotated Logic Programs [11], Fuzzy Logic Programming [17], Hybrid Probabilistic Logic Programs [5], and Possibilistic Logic Programming [**?**]. In [14], a propositional approach considering the multi-adjoint philosophy was studied. Later, a first-order extension arose [13] as a generalization of the propositional one, which enables to deal with more complex sentences and reasonings. For example, it allows the use of a huge variety of different adjoint pairs in the construction of the rules of a logic program. Consequently, a much more flexible framework is achieved for the processing of information and the modeling of different scenarios.

The immediate consequences operator is essential to define the semantics of a logic program [12]. Indeed, it is basic for the fixed-point semantics, in which models of a program are characterized through the post-fixed points of this operator and the least model of the program can be obtained by iterating the immediate consequences operator an ordinal number of times, starting from the least interpretation.

The use of substitution mappings (or simply substitutions) is also a fundamental notion in first-order logic [7,**?**,16]. The definition of an arbitrary interpretation on general formulas considers substitutions; the matching of formulas also

* Partially supported by the project PID2019-108991GB-I00 funded by MICIU/AEI/10.13039/501100011033, the project PID2022-137620NB-I00 funded by MICIU/AEI/10.13039/501100011033 and FEDER, UE, by the grant TED2021-129748B-I00 funded by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR, and by the research and transfer program of the University of Cádiz in project PR2022-057

needs substitutions (unifiers); and the definition of the immediate consequences operator on first-order logic [16] requires substitutions too, among others. However, the kind of these substitutions in this last definition and the proper definition of immediate consequences operator need diverse clarifications. This paper will prove that the substitutions cannot be arbitrary ones, but ground in the variables of the rules. Moreover, we will show that the definition of the immediate consequences operator is based on the computation of two suprema on two notable sets. These clarifications will be essential in future extensions and advances in residuated and multi-adjoint first-order logic programming approaches.

The structure of the paper is as follows. In Section 2 some preliminaries results about multi-adjoint algebra and first-order multi-adjoint logic programming will be recalled. Section 3 presents the main contribution of the paper with more clarified definitions of the immediate consequences operator, together with the characterization of the models as post-fixed points of the immediate consequence operator and an illustrative example. Finally, in Section 4 some conclusions and future work are given.

2 Preliminaries

In this section we will present the algebraic structure that we are going to use in the paper and we will recall the main notions of multi-adjoint logic programming.

2.1 Multi-adjoint algebra

The considered operators in this algebra are the adjoint pairs. Each adjoint pair is composed of a conjunctor and an implication, which is a generalization of a t-norm and its residuated implication. We will write the operators defined on an algebraic structure (poset or lattice) with a dot, and their symbols in a graded set without dot.

Definition 1 ([16]). *Let $\langle P, \preceq\rangle$ be a poset and $(\dot{\leftarrow}, \dot{\&})$ a pair of binary operators in P such that*

1. *The operator $\dot{\&}$ is increasing in both arguments.*
2. *The operator $\dot{\leftarrow}$ is increasing in the first argument (*consequent*) and decreasing in the second one (*antecedent*)*
3. *The adjoint property holds, i.e., $x \preceq (z \dot{\leftarrow} y)$ if and only if $(x \dot{\&} y) \preceq z$, for all $x, y, z \in P$.*

Then, we say that $(\dot{\leftarrow}, \dot{\&})$ forms an adjoint pair *in $\langle P, \preceq\rangle$.*

Now, the definitions of graded set and multi-adjoint lattice are recalled.

Definition 2 ([14]). *A* graded set *is a set Ω with a function which assigns to each element $\omega \in \Omega$ a number $n \geq 0$, called the* arity *of ω.*

Definition 3 ([14]). *Let $\langle L, \preceq\rangle$ be a complete lattice. A* multi-adjoint lattice $\mathcal{L}$ *is a tuple $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ satisfying*

1. $\langle L, \preceq \rangle$ *is bounded, i.e., it has a bottom* $\bot$ *and a top* $\top$ *elements.*
2. $(\dot{\leftarrow}_i, \dot{\&}_i)$ *is an adjoint pair in* $\langle L, \preceq \rangle$*, for all* $i \in \{1, \ldots, n\}$*.*
3. $\top \dot{\&}_i \vartheta = \vartheta \dot{\&}_i \top = \vartheta$*, for all* $\vartheta \in L$ *and* $i \in \{1, \ldots, n\}$*.*

Next, we recall the definition of multi-adjoint Ω-algebra, where the definition of Ω-algebra can be consulted in [14].

Definition 4 ([16]). *Let* Ω *be a graded set containing the operator symbols* $\leftarrow_i$ *and* $\&_i$*, for* $i \in \{1, \ldots, n\}$*, and possibly some extra operators, and let* $\mathfrak{L}$ *be an* Ω*-algebra on the complete lattice* $\langle L, \preceq \rangle$*. We say that* $\mathfrak{L}$ *is a* multi-adjoint Ω-algebra *with respect to the pairs* $(\leftarrow_i, \&_i)$*, for* $i \in \{1, \ldots, n\}$*, if* $\mathcal{L} = (L, \preceq, \dot{\leftarrow}_1, \dot{\&}_1, \ldots, \dot{\leftarrow}_n, \dot{\&}_n)$ *is a multi-adjoint lattice.*

Now, an example of multi-adjoint Ω-algebra is introduced.

Example 1. Let $\Omega = \left\{\leftarrow_{\mathrm{P}}, \&_{\mathrm{P}}, \leftarrow_{\mathrm{G}}, \&_{\mathrm{G}}, \wedge_{\mathrm{L}}, @_{(2,1)}\right\}$ be a graded set and $\mathfrak{L} = \langle [0,1], I \rangle$ an Ω-algebra, where $\dot{\&}_{\mathrm{P}}$ and $\dot{\&}_{\mathrm{G}}$ are the product and Gödel t-norms respectively, together with their residuated implications. In addition, we are considering the Łukasiewicz conjunctor $\dot{\wedge}_{\mathrm{L}}$ and the aggregator $\dot{@}_{(2,1)}$. It is easy to prove that $\mathcal{L} = ([0,1], \leq, \dot{\leftarrow}_{\mathrm{P}}, \dot{\&}_{\mathrm{P}}, \dot{\leftarrow}_{\mathrm{G}}, \dot{\&}_{\mathrm{G}})$ is a multi-adjoint lattice, and then $\mathfrak{L}$ is a multi-adjoint Ω-algebra. These operators are defined as

$$\dot{\&}_{\mathrm{G}}(x, y) = \min\{x, y\} \qquad z \dot{\leftarrow}_{\mathrm{G}} y = \begin{cases} 1 & \text{if} \quad y \leq z \\ z & \text{otherwise} \end{cases}$$

$$\dot{\&}_{\mathrm{P}}(x, y) = x \cdot y \qquad z \dot{\leftarrow}_{\mathrm{P}} y = \begin{cases} 1 & \text{if} \quad y \leq z \\ \frac{z}{y} & \text{otherwise} \end{cases}$$

$$\dot{\wedge}_{\mathrm{L}}(x, y) = \max\{0, x + y - 1\} \qquad \dot{@}_{(2,1)}(x, y) = \frac{2x + y}{3}$$

for all x, y, $z \in [0,1]$. □

2.2 Multi-adjoint logic programming

First of all, we will present the syntax of multi-adjoint logic programming. Diverse relevant notions such as alphabet, language and well-formed formulas can be consulted in [12,16]. Next, in order to fix the notation and the starting point, we recall the definition of logic program, which is a set of rules and facts in a considered first-order extended language $\mathfrak{F}_e$, which is the corresponding Ω-algebra of formulas freely generated from the disjoint union of the set of the symbols from the first-order language and the lattice [9,16]. From now on, a graded set Ω containing conjunctor, disjunctor and aggregator symbols, a multi-adjoint lattice $\mathcal{L} = (L, \preceq, \dot{\leftarrow}_1, \dot{\&}_1, \ldots, \dot{\leftarrow}_n, \dot{\&}_n)$ and its Ω-algebra $\mathfrak{L}$ will be fixed.

Definition 5 ([16]). *A* multi-adjoint logic program *is a set* $\mathbb{P}$ *of weighted rules of the form* $\langle H \leftarrow_i \mathcal{B}; \vartheta \rangle$*, also denoted as* $H \overset{\vartheta}{\leftarrow}_i \mathcal{B}$*, such that:*

1. *The consequent of the implication, H, is an atom called the* head.
2. *The antecedent of the implication, $\mathcal{B}$, is called the* body*, and it is a formula built from atoms $B_1, \ldots, B_n$, with $0 \leq n$, by the use of conjunctors, disjunctors and aggregators.*
3. *The* confidence factor *ϑ is an element, a truth-value of L.*

A program is completed with particular observations in the form of facts*, which are rules with body $\top$. Moreover, free occurrences of variables in the program are considered to be universally quantified.*

Note that the $\top$ element in the body of a fact is considered a constant of the first-order language. Furthermore, the formula $\mathcal{B}$ of the body of a rule can be written from its more basic elements as $@\,[B_1, \ldots, B_n]$, where $@$ is the aggregation of the symbols of operators appearing in $\mathcal{B}$, and $B_1, \ldots, B_n$ are the atoms of $\mathcal{B}$. Next, we present an example of logic program, which will be used later in the next section.

Example 2. Consider the multi-adjoint Ω-algebra $\mathfrak{L}$ introduced in Example 1, where each operator symbol ω in the graded set Ω has been associated with its respective interpretation $\dot{\omega}$, with the aim of simplifying the notation. With the algebraic structure fixed, we can consider the following program $\mathbb{P}$.

$$\begin{aligned}
\mathcal{R}_1\colon &\quad p(a,Y) \overset{0.8}{\leftarrow}_{\mathrm{G}} && @_{(2,1)}\,(q(X) \&_{\mathrm{G}} r(Y), s(X,b))\\
\mathcal{R}_2\colon &\quad p(X,b) \overset{0.9}{\leftarrow}_{\mathrm{P}} && t(X) \wedge_{\mathrm{L}} r(Y)\\
\mathcal{R}_3\colon &\quad q(X) \overset{0.2}{\leftarrow}_{\mathrm{P}} && t(X)
\end{aligned}$$

In each particular case, specific values are "measured" to some variables. These variables are called *hypotheses* and are formally expressed as facts. For example, we can consider the following facts associated with the ground atoms $r(b)$, $s(a,b)$ and $t(a)$,

$$\begin{aligned}
\mathcal{R}_4\colon &\quad r(b) \overset{0.6}{\leftarrow} && \top\\
\mathcal{R}_5\colon &\quad s(a,b) \overset{0.8}{\leftarrow} && \top\\
\mathcal{R}_6\colon &\quad t(a) \overset{0.6}{\leftarrow} && \top
\end{aligned}$$

The set composed of the previous rules $\mathcal{R}_1$, ..., $\mathcal{R}_6$ is an example of multi-adjoint logic program. □

Now, the semantics of first-order logic programming will be recalled. The following notion provides a meaning (representative truth-value) to syntactic formulas of the logic framework.

Definition 6 ([16]). *Let $\mathbb{P}$ be a multi-adjoint logic program. An* interpretation *is a mapping from the set of ground atoms to the lattice of truth-values $\langle L, \preceq \rangle$. The set of all interpretations of the formulas defined by the Ω-algebra $\mathfrak{F}$ in the Ω-algebra $\mathfrak{L}$ is denoted as $\mathcal{I}_{\mathfrak{L}}$.*

Although interpretations are defined for ground atoms, thanks to the homomorphic extension theorem [14], we can extend the definition of an interpretation I for all ground formulas of the language, what we will denote as $\hat{I}$. Besides, taking into account that all formulas are universally closed, we can extend the definition of an interpretation I for a non-ground formula A as follows.

$$\hat{I}(A) = \inf\{\hat{I}(A\xi) \mid A\xi \text{ is a ground instantiation of } A\}$$

The main notion of the semantics in logic programming is the definition of model.

Definition 7 ([16]). *Given an interpretation $I \in \mathcal{I}_{\mathfrak{L}}$, a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ is* satisfied *by I if and only if $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$. An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a* model *of a multi-adjoint logic program $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.*

3 Analyzing immediate consequences operator

Fixed-point semantics was studied in the Boolean and fuzzy settings [12,16] in order to compute the least model of a logic program, which provides the most relevant information that it is possible to extract from the program, and can be used for instance in automatic recommendation systems.

This special semantics is based on the immediate consequences operator. This operator aggregates, for every ground atom A, the information given by each rule with head H satisfying that A and H unify. Notice that, given a ground substitution ξ and a rule $H \leftarrow_i \mathcal{B}$, we obtain the following chain of equalities

$$\hat{I}((H \leftarrow_i \mathcal{B})\xi) = \hat{I}(H\xi \leftarrow_i \mathcal{B}\xi) = \hat{I}(H\xi) \dot{\leftarrow}_i \hat{I}(\mathcal{B}\xi) = I(H\xi) \dot{\leftarrow}_i \hat{I}(\mathcal{B}\xi)$$

As a consequence, the operator $T_{\mathbb{P}} \colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$ defined as

$$T_{\mathbb{P}}(I)(A) = \sup\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}\theta) \mid \langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P} \text{ and } A = C\theta\right\} \tag{1}$$

for all interpretation I and ground atom A, was introduced in [16], as a generalization of the Boolean immediate consequences operator given by van Emden and Kowalski in [6].

However, in Expression (1) it is unclear whether the substitution θ should be ground in some sense or not. The following example shows that an arbitrary non-ground substitution cannot be considered in the previous expression, since in this case the models are not characterized by the fixed-points of $T_{\mathbb{P}}$, as it happens in the classical case.

Example 3. Consider the program $\mathbb{P}$ introduced in Example 2 and the interpretation I defined on the ground atoms as

$$\begin{array}{rrr} I(r(a)) = 0.1 & I(r(b)) = 0.6 & I(s(a,b)) = 0.8 \\ I(s(b,b)) = 0.1 & I(t(a)) = 0.6 & I(t(b)) = 0.1 \\ I(q(a)) = 0.12 & I(q(b)) = 0.02 & I(p(a,a)) = 0.1 \\ I(p(b,b)) = 0.1 & I(p(a,b)) = 0.1 & \end{array}$$

Furthermore, we can extend this definition for non-ground atoms, following Definition 6, as follows.

$$\begin{aligned}\hat{I}(q(X)) &= \inf\{I(q(a)), I(q(b))\} = \inf\{0.12, 0.02\} = 0.02\\ \hat{I}(s(X,b)) &= \inf\{I(s(a,b)), I(s(b,b))\} = \inf\{0.8, 0.1\} = 0.1\\ \hat{I}(r(Y)) &= \inf\{I(r(a)), I(r(b))\} = \inf\{0.1, 0.6\} = 0.1\end{aligned}$$

First of all, we will see that I satisfies that $T_{\mathbb{P}}(I) \sqsubseteq I$. We will start proving it for $p(a,b)$. For this, we need to compute the value $T_{\mathbb{P}}(I)(p(a,b))$. According to Expression (1), we consider the substitutions $\theta_1 = \{X/X', Y/b\}$[1] applied to $\mathcal{R}_1$; and $\theta_2 = \{X/a, Y/Y'\}$ applied to the second rule $\mathcal{R}_2$.

$$\begin{aligned}T_{\mathbb{P}}(I)(p(a,b)) &= \sup\left\{0.8\dot{\&}_{\mathrm{G}}\hat{I}(\mathcal{B}_1\theta_1), 0.9\dot{\&}_{\mathrm{P}}\hat{I}(\mathcal{B}_2\theta_2)\right\}\\ &= \sup\Big\{0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\left(\hat{I}\left(q\left(X'\right)\right)\dot{\&}_{\mathrm{G}}I\left(r\left(b\right)\right), \hat{I}\left(s\left(X',b\right)\right)\right),\\ &\qquad 0.9\dot{\&}_{\mathrm{P}}\left(I\left(t\left(a\right)\right)\dot{\wedge}_{\mathrm{L}}\hat{I}\left(r\left(Y'\right)\right)\right)\Big\}\\ &= \sup\left\{0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}(0.02\dot{\&}_{\mathrm{G}}0.6, 0.1), 0.9\dot{\&}_{\mathrm{P}}\left(0.6\dot{\wedge}_{\mathrm{L}}0.1\right)\right\}\\ &= \sup\{0.047, 0\}\\ &\leq 0.1 = I(p(a,b))\end{aligned}$$

It is easy to prove the inequality $T_{\mathbb{P}}(I) \sqsubseteq I$ for the rest of the ground atoms, as we can see below.

$$\begin{aligned}T_{\mathbb{P}}(I)(r(a)) &= 0 \leq I(r(a)) & T_{\mathbb{P}}(I)(r(b)) &= 0.6 \leq I(r(b))\\ T_{\mathbb{P}}(I)(s(a,b)) &= 0.8 \leq I(s(a,b)) & T_{\mathbb{P}}(I)(s(b,b)) &= 0 \leq I(s(b,b))\\ T_{\mathbb{P}}(I)(t(a)) &= 0.6 \leq I(t(a)) & T_{\mathbb{P}}(I)(t(b)) &= 0 \leq I(t(b))\\ T_{\mathbb{P}}(I)(q(a)) &= 0.12 \leq I(q(a)) & T_{\mathbb{P}}(I)(q(b)) &= 0.02 \leq I(q(b))\\ T_{\mathbb{P}}(I)(p(a,a)) &= 0.046 \leq I(p(a,a)) & T_{\mathbb{P}}(I)(p(b,b)) &= 0 \leq I(p(b,b))\end{aligned}$$

Now, we are going to see that I is not a model for $\mathbb{P}$. Following Definition 2, we have that I is a model for $\mathbb{P}$ if, for every rule $\langle A \leftarrow_i \mathcal{B}_i\rangle \in \mathbb{P}$, the following inequality holds.

$$\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B}_i) = \inf\{\hat{I}((A \leftarrow_i \mathcal{B}_i)\xi) \mid \xi \text{ provides a ground instantiation}\} \quad (2)$$

Notice that Expression (2) must be satisfied for every substitution ξ satisfying that $(A \leftarrow_i \mathcal{B}_i)\xi$ is a ground instantiation and, in particular, it should be satisfied if we consider $\xi = \{X/a, Y/b\}$ and the rule $\mathcal{R}_2 = \langle p(X,b) \leftarrow_{\mathrm{P}} t(X) \wedge_{\mathrm{L}} r(Y); 0.9\rangle$.

[1] Notice that the substitution θ_1 interchanges in the formula the variable X by the variable X' and Y by b.

Nevertheless, the following chain of inequalities arises.

$$\begin{aligned}\hat{I}\left((p(X,b) \leftarrow_{\mathrm{P}} t(X) \wedge_{\mathrm{L}} r(Y))\xi\right) &= \hat{I}((p(a,b) \leftarrow_{\mathrm{P}} t(a) \wedge_{\mathrm{L}} r(b))) \\ &= I(p(a,b)) \dot{\leftarrow}_{\mathrm{P}} (I(t(a)) \dot{\wedge}_{\mathrm{L}} I(r(b))) \\ &= 0.1 \dot{\leftarrow}_{\mathrm{P}} (0.6 \dot{\wedge}_{\mathrm{L}} 0.6) \\ &= 0.5 \\ &\ngeq 0.9\end{aligned}$$

Therefore, Expression (2) does not hold and we have found an interpretation I, such that $T_{\mathbb{P}}(I) \sqsubseteq I$, but I is not a model for $\mathbb{P}$.

Thus, we have obtained that the characterization of models based on the post-fixed points of the immediate consequences operator does not hold. □

As a consequence, it is clear that the substitution θ must provide a ground instantiation of the rule in Expression (1). The following definition clarifies this fact in the definition of immediate consequences operator given in [16].

Definition 8. *The* immediate consequences operator $T_{\mathbb{P}}$ *maps interpretations to interpretations and it is defined in such a way that, given an interpretation I and a ground atom A, $T_{\mathbb{P}}(I)(A)$ is computed as*

$$\sup\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}\xi) \mid \langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}, A = C\xi, \text{ and } (C \leftarrow_i \mathcal{B})\xi \text{ is ground}\right\}$$

The set of variables in a rule $\mathcal{R}$ will be denoted as $\mathcal{V}(\mathcal{R})$. Hence, the last property in the definition of $T_{\mathbb{P}}(I)(A)$ can be written as: "ξ is a substitution ground in $\mathcal{V}(\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle)$". The following result provides a different point of view of the definition of the $T_{\mathbb{P}}$ operator. We need to remark that Definition 8 really computes two suprema, one on the substitutions and the other one on the rules of the program. Computationally, it does not change, but it is interesting for studying more properties and future generalizations of the operator.

Theorem 1. *Let $\mathbb{P}$ be a program, an interpretation I and a ground atom A. We obtain that*

$$T_{\mathbb{P}}(I)(A) = \sup\left\{S_I(\vartheta \&_i \mathcal{B}\theta) \mid \langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}, \text{ and } \theta = mgu\{A, C\}\right\}$$

where the mapping $S_I \colon \mathfrak{F}_e \to L$ is defined for every $\phi \in \mathfrak{F}_e$ as $S_I(\phi) = \sup\{\hat{I}(\phi\xi) \mid$ ξ is a ground substitution$\}$.

Proof. Given a ground atom A and an interpretation I, for each rule $\mathcal{R} = \langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}$, such that there exists $\theta = \mathrm{mgu}\{A, C\}$, we must compute in Definition 8 the value $\sup\left\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}\theta\xi) \mid \xi \text{ is a ground substitution}\right\}$, which is given by the mapping $S_I \colon \mathfrak{F} \to L$. Finally, the computation of $T_{\mathbb{P}}(I)(A)$ arises as the supremum of this value, $S_I(\vartheta \&_i \mathcal{B}\theta)$, for all such rules $\mathcal{R} = \langle C \leftarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}$, satisfying that there exists $\theta = \mathrm{mgu}\{A, C\}$. □

Notice that it is not necessary for θ to be a ground substitution. This last definition is taken into consideration to assert that this operator just allows the characterization of models of a given program $\mathbb{P}$ through the post-fixed points of the immediate consequences operator, which also clarifies the proof given in [16].

Theorem 2. *An interpretation I is a model of a multi-adjoint logic program $\mathbb{P}$ if and only if $T_{\mathbb{P}}(I) \sqsubseteq I$.*

Proof. Given a model I for $\mathbb{P}$ and a ground atom A, we will prove that $T_{\mathbb{P}}(I) \sqsubseteq I$. On the one hand, if there is no rule in $\mathbb{P}$ matching with A, then $T_{\mathbb{P}}(I)(A) = \sup \varnothing = \bot \preceq I(A)$, and the result is trivially obtained.

On the other hand, consider an arbitrary rule $\mathcal{R} = \langle C \leftarrow_i \mathcal{B}, \vartheta \rangle$ of $\mathbb{P}$, such as there exists $\theta = \mathrm{mgu}\{A, C\}$. Moreover, given a ground substitution ξ, we have that $\theta\xi$ is also ground and, by hypothesis, the following chain of inequalities holds, where σ denotes a substitution.

$$\begin{aligned}
\vartheta &\overset{(*)}{\preceq} \hat{I}\,(C \leftarrow_i \mathcal{B}) \\
&= \inf\{\hat{I}\,((C \leftarrow_i \mathcal{B})\sigma) \mid (C \leftarrow_i \mathcal{B})\sigma \text{ is a ground instantiation of } C \leftarrow_i \mathcal{B}\} \\
&\preceq \hat{I}\,((C \leftarrow_i \mathcal{B})\theta\xi) \\
&= \hat{I}\,(C\theta \leftarrow_i \mathcal{B}\theta\xi) \\
&= \hat{I}(C\theta) \dot{\leftarrow}_i \hat{I}(\mathcal{B}\theta\xi)
\end{aligned}$$

where $(*)$ holds due to the fact that I is a model for $\mathbb{P}$. Applying the adjoint property, we can write

$$\hat{I}((\vartheta \&_i \mathcal{B}\theta)\xi) = \vartheta \dot{\&}_i \hat{I}(\mathcal{B}\theta\xi) \preceq \hat{I}(C\theta\xi) = I(A\xi) = I(A)$$

where the last equality holds because the atom A is ground. Therefore, taking suprema on the set of ground substitutions, we reach

$$S_I(\vartheta \&_i \mathcal{B}\theta) = \sup\{\hat{I}((\vartheta \&_i \mathcal{B}\theta)\xi) \mid \xi \text{ is a ground substitution}\} \preceq I(A)$$

Now, taking suprema on the set of rules such that there exists $\theta = \mathrm{mgu}\{A, C\}$, the following inequality arises.

$$\begin{aligned}
T_{\mathbb{P}}(I)(A) &= \sup\{S_I(\vartheta \&_i \mathcal{B}\theta) \mid \langle C \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}, \text{ and } \theta = \mathrm{mgu}\{A, C\}\} \\
&\preceq I(A)
\end{aligned}$$

Consequently, the inequality $T_{\mathbb{P}}(I) \sqsubseteq I$ holds. Now, we consider an interpretation I satisfying that $T_{\mathbb{P}}(I) \sqsubseteq I$, and we will see that I is a model for $\mathbb{P}$. For this, we consider an arbitrary weighted rule $\mathcal{R} = \langle C \leftarrow_i \mathcal{B}, \vartheta \rangle$ in $\mathbb{P}$ and we need to prove that

$$\begin{aligned}
&\vartheta \preceq \hat{I}(C \leftarrow_i \mathcal{B}) \\
&\quad = \inf\{\hat{I}((C \leftarrow_i \mathcal{B})\sigma) \mid (C \leftarrow_i \mathcal{B})\sigma \text{ is a ground instantiation of } C \leftarrow_i \mathcal{B}\}
\end{aligned}$$

that is, $\vartheta \preceq \hat{I}((C \leftarrow_i \mathcal{B})\sigma)$ for all substitution σ such that $(C \leftarrow_i \mathcal{B})\sigma$ is a ground instantiation of $C \leftarrow_i \mathcal{B}$.

Hence, given one of these substitutions σ and a ground substitution ξ, applying that $T_{\mathbb{P}}(I) \sqsubseteq I$, we have that

$$\vartheta \dot{\&}_i \hat{I}(\mathcal{B}\sigma) = \vartheta \dot{\&}_i \hat{I}(\mathcal{B}\sigma\xi) = \hat{I}((\vartheta \&_i \mathcal{B}\sigma)\xi) \preceq S_I(\vartheta \&_i \mathcal{B}\sigma) \overset{(*)}{\preceq} T_{\mathbb{P}}(I)(C\sigma) \preceq I(C\sigma)$$

where $(*)$ follows from the definition of $T_{\mathbb{P}}$. Now, by the adjoint property, we obtain $\vartheta \preceq I(C\sigma) \dot{\leftarrow}_i \hat{I}(\mathcal{B}\sigma) = \hat{I}((C \leftarrow_i \mathcal{B})\sigma)$ and finally, taking infima on the substitutions σ, we obtain the required inequality. □

Notice that we cannot ensure the following expressions are equal:

$$\sup\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}\theta\xi) \mid \xi \text{ is a ground substitution}\} \tag{3}$$
$$\vartheta \dot{\&}_i \sup\{\hat{I}(\mathcal{B}\theta\xi) \mid \xi \text{ is a ground substitution}\} \tag{4}$$

because of the operator $\dot{\&}_i$ could not preserve the supremum in the second argument (see for example [1,2,3]). This fact caused the need to consider the extended language $\mathfrak{F}_e$, including the truth-values in the language, due to $\hat{I}(\mathcal{B}\xi)$ is an element of L. With the following example, we will show how the value of the immediate consequences operator is obtained according to Theorem 1.

Example 4. Consider Example 2 and the interpretation J given by

$$\begin{array}{lll} J(r(a)) = 0 & J(r(b)) = 0.6 & J(s(a,b)) = 0.8 \\ J(s(b,b)) = 1 & J(t(a)) = 0.6 & J(t(b)) = 0 \\ J(q(a)) = 1 & J(q(b)) = 1 & \end{array}$$

We are going to compute the value $T_{\mathbb{P}}(J)(p(a,b))$ considering the substitutions $\theta_1 = \{X/X', Y/b\}$ and $\theta_2 = \{X/a, Y/Y'\}$, which are ground on the variables of rules $\mathcal{R}_1$ and $\mathcal{R}_2$.

$$\begin{aligned} T_{\mathbb{P}}(J)(p(a,b)) &= \sup\{S_J(0.8 \&_{\mathrm{G}} \mathcal{B}_1\theta_1), S_J(0.9 \&_{\mathrm{P}} (\mathcal{B}_2\theta_2))\} \\ &= \sup\big\{S_J\big(0.8 \&_{\mathrm{G}} \big(@_{(2,1)}(q(X) \&_{\mathrm{G}} r(Y), s(X,b))\,\theta_1\big)\big), \\ &\qquad S_J(0.9 \&_{\mathrm{P}} ((t(X) \wedge_{\mathrm{L}} r(Y))\,\theta_2))\big\} \\ &= \sup\big\{S_J\big(0.8 \&_{\mathrm{G}} @_{(2,1)}(q(X') \&_{\mathrm{G}} r(b), s(X',b))\big), \\ &\qquad S_J(0.9 \&_{\mathrm{P}} (t(a) \wedge_{\mathrm{L}} r(Y')))\big\} \end{aligned}$$

$$
\begin{aligned}
&= \sup\Big\{\sup\Big\{0.8\dot{\&}_{\mathrm{G}}\hat{J}\left(@_{(2,1)}\left(q(a)\&_{\mathrm{G}}r(b), s(a,b)\right)\right),\\
&\qquad 0.8\dot{\&}_{\mathrm{G}}\hat{J}\left(@_{(2,1)}\left(q(b)\&_{\mathrm{G}}r(b), s(b,b)\right)\right)\Big\},\\
&\qquad \sup\Big\{0.9\dot{\&}_{\mathrm{P}}\hat{J}\left(t(a)\wedge_{\mathrm{L}} r(a)\right), 0.9\dot{\&}_{\mathrm{P}}\hat{J}\left(t(a)\wedge_{\mathrm{L}} r(b)\right)\Big\}\Big\}\\
&= \sup\Big\{\sup\Big\{0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\Big(J(q(a))\dot{\&}_{\mathrm{G}}J(r(b)), J(s(a,b))\Big),\\
&\qquad 0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\Big(J(q(b))\dot{\&}_{\mathrm{G}}J(r(b)), J(s(b,b))\Big)\Big\},\\
&\qquad \sup\Big\{0.9\dot{\&}_{\mathrm{P}}\left(J(t(a))\dot{\wedge}_{\mathrm{L}}J(r(a))\right), 0.9\dot{\&}_{\mathrm{P}}\left(J(t(a))\dot{\wedge}_{\mathrm{L}}J(r(b))\right)\Big\}\Big\}\\
&= \sup\Big\{\sup\Big\{0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\Big(1\dot{\&}_{\mathrm{G}}0.6, 0.8\Big), 0.8\dot{\&}_{\mathrm{G}}\dot{@}_{(2,1)}\Big(1\dot{\&}_{\mathrm{G}}0.6, 1\Big)\Big\},\\
&\qquad \sup\Big\{0.9\dot{\&}_{\mathrm{P}}\left(0.6\dot{\wedge}_{\mathrm{L}}0\right), 0.9\dot{\&}_{\mathrm{P}}\left(0.6\dot{\wedge}_{\mathrm{L}}0.6\right)\Big\}\Big\}\\
&= \sup\left\{\sup\left\{0.67, 0.73\right\}, \sup\left\{0, 0.18\right\}\right\}\\
&= 0.73
\end{aligned}
$$

□

4 Conclusions and future work

In this paper, we have revised the notion of the immediate consequences operator given in [16] and we have clarified the use of ground substitutions in its definition. This clarification ensures the characterization of the models for a program $\mathbb{P}$ as the post-fixed points of the immediate consequences operator, i.e., $T_{\mathbb{P}}(I) \sqsubseteq I$. Furthermore, we have remarked the use of two suprema in the definition of the immediate consequences operator, which will be fundamental in future extensions and studies of this operator. For example, a procedural semantics [15] based on reductants [8,9,10] will be analyzed and the approaches based on generalized quantifiers [**?**,**?**] and ordered weighted averaging (OWA) operators [**?**] will be studied in the first-order setting. Moreover, the comparison with recent approaches, such as fuzzy order-sorted feature logic [**?**], will be discussed.

References

1. M. E. Cornejo, J. Medina, and E. Ramírez-Poussa. A comparative study of adjoint triples. *Fuzzy Sets and Systems*, 211:1–14, 2013.
2. M. E. Cornejo, J. Medina, and E. Ramírez-Poussa. Multi-adjoint algebras versus non-commutative residuated structures. *International Journal of Approximate Reasoning*, 66:119–138, 2015.
3. M. E. Cornejo, J. Medina, and E. Ramírez-Poussa. Adjoint negations, more than residuated negations. *Information Sciences*, 345:355 – 371, 2016.
4. C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, pages 748–759. Lecture Notes in Artificial Intelligence, 2143, 2001.

5. A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *Journal of Logic Programming*, 43:187–250, 2000.
6. M. v. Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.
7. F. Formato, G. Gerla, and M. Sessa. Similarity-based unification. *Fundamenta Informaticae*, 41(4):393–414, 2000.
8. P. Julián, G. Moreno, and J. Penabad. An improved reductant calculus using fuzzy partial evaluation techniques. *Fuzzy Sets and Systems*, 160:162–181, 2009.
9. P. Julián-Iranzo, J. Medina, and M. Ojeda-Aciego. On reductants in the framework of multi-adjoint logic programming. *Fuzzy Sets and Systems*, 317:27 – 43, 2017.
10. P. Julián-Iranzo, J. Medina, and M. Ojeda-Aciego. Revisiting reductants in the multi-adjoint logic programming framework. *Lecture Notes in Artificial Intelligence*, 8761:694–702, 2014.
11. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12:335–367, 1992.
12. J. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.
13. J. Medina and M. Ojeda-Aciego. On first-order multi-adjoint logic programming. In *XI Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2002; León (España)*, 2002.
14. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. In *Logic Programming and Non-Monotonic Reasoning, LPNMR'01*, pages 351–364. Lecture Notes in Artificial Intelligence 2173, 2001.
15. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programming. In *Progress in Artificial Intelligence, EPIA'01*, pages 290–297. Lecture Notes in Artificial Intelligence 2258, 2001.
16. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 146:43–62, 2004.
17. P. Vojtáš. Fuzzy logic programming. *Fuzzy sets and systems*, 124(3):361–370, 2001.

Bibliografía

[1] N. U. Ahamed, Z. B. Taha, I. B. M. Khairuddin, M. Rabbi, S. M. Rahaman y K. Sundaraj. Fuzzy logic controller design for intelligent air-conditioning system. En *2016 2nd International Conference on Control Science and Systems Engineering (ICCSSE)*, páginas 232–236, 2016.

[2] B. Ahn y H. Park. An efficient pruning method for decision alternatives of OWA operators. *IEEE Transactions on Fuzzy Systems*, 16:1542–1549, 2009.

[3] H. S. Ali, R. R. Rout, P. Parimi y S. K. Das. Real-time task scheduling in fog-cloud computing framework for IoT applications: A fuzzy logic based approach. En *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, páginas 556–564, 2021.

[4] Z. T. Allawi y T. Y. Abdalla. A PSO-optimized type-2 fuzzy logic controller for navigation of multiple mobile robots. En *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*, páginas 33–39, 2014.

[5] J. Almendros-Jimenez, A. Becerra-Terón, G. Moreno y J. Riaza. Tuning fuzzy SPARQL queries in a fuzzy logic programming environment. En *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1–7, 2019.

[6] J. M. Almendros-Jiménez, M. Bofill, A. Luna-Tedesqui, G. Moreno, C. Vázquez y M. Villaret. Fuzzy XPath for the automatic search of fuzzy formulae models. En C. Beierle y A. Dekhtyar, editores, *Scalable Uncertainty Management*, páginas 385–398. Springer International Publishing, Cham, 2015.

[7] J. M. Almendros-Jiménez, A. Becerra-Terón y G. Moreno. Fuzzy queries of social networks with FSA-SPARQL. *Expert Systems with Applications*, 113:128–146, 2018.

[8] R. G. Aragón, M. E. Cornejo, J. Medina, J. Moreno-García y E. Ramírez-Poussa. Decision support system for photovoltaic fault detection avoiding meteorological conditions. *International Journal of Information Technology & Decision Making*, 21(03):911–932, 2022.

[9] K. J. Arrow. *Social Choice and Individual Values*. Yale University Press, 2012.

[10] A. Azadegan, L. Porobic, S. Ghazinoory, P. Samouei y A. Saman Kheirkhah. Fuzzy logic in manufacturing: A review of literature and a specialized application. *International Journal of Production Economics*, 132(2):258–270, 2011.

[11] J. W. de Bakker. *Recursive procedures*. Mathematical Centre Tracts, 24, Mathematisch Centrum, Ámsterdam, 1971.

[12] P. Balaji y D. Srinivasan. Type-2 fuzzy logic based urban traffic management. *Engineering Applications of Artificial Intelligence*, 24(1):12–22, 2011.

[13] J. Barwise y R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219, 1981.

[14] C. Batini y M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications)*. Springer-Verlag, Berlín, Heidelberg, 2006.

[15] R. E. Bellman y L. A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):B141–B164, 1970.

[16] J. M. Belman-Flores, D. A. Rodríguez-Valderrama, S. Ledesma, J. J. García-Pabón, D. Hernández y D. M. Pardo-Cely. A review on applications of fuzzy logic control for refrigeration systems. *Applied Sciences*, 12(3), 2022.

[17] J. Bernoulli. *Ars conjectandi*. Impensis Thurnisiorum, fratrum, 1713.

[18] M. Biglarbegian, W. W. Melek y J. M. Mendel. Design of novel interval type-2 fuzzy controllers for modular and reconfigurable robots: Theory and experiments. *IEEE Transactions on Industrial Electronics*, 58(4):1371–1384, 2011.

[19] J. M. Bocheński. *A History of Formal Logic*. University of Notre Dame Press, 1961.

[20] G. Bojadziev y M. Bojadziev. *Fuzzy Logic for Business, Finance, and Management*. World Scientific Publishing Co., Inc., EE.UU., 2nd edition, 2007.

[21] G. Boole. *An Investigation of the Laws of Thought*. Dover Publications, Nueva York, 1854.

[22] G. Boole. *The Mathematical Analysis of Logic: Being an Essay Towards a Calculus of Deductive Reasoning*. Cambridge Library Collection - Mathematics. Cambridge University Press, 2009.

[23] P. Bosc, L. Lietard y O. Pivert. Flexible database querying and the division of fuzzy relations. *Scientia Iranica*, 2(4):329–340, 1996.

[24] T. Calvo, G. Mayor y R. Mesiar, editors. *Aggregation operators: new trends and applications*. Physica-Verlag GmbH, DEU, 2002.

[25] M. Calvo-Flores, D. Sánchez, M. Vila y J. Serrano. A survey of methods to evaluate quantified sentences. *Mathware and soft computing*, 7(2–3):149–158, 2000.

[26] G. Cantor. Ueber eine eigenschaft des inbegriffs aller reellen algebraischen zahlen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 77:258–262, 1874.

[27] R. Carnap. *Logical Foundations of Probability*. Chicago University of Chicago Press, Chicago, 1950.

[28] O. Castillo, H. Neyoy, J. Soria, M. García y F. Valdez. Dynamic fuzzy logic parameter tuning for ACO and its application in the fuzzy logic control of an autonomous mobile robot. *International Journal of Advanced Robotic Systems*, 10:51, 2013.

[29] E. Çakir y Z. Ulukan. A fuzzy linguistic programming for sustainable ecotourism activities. En *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, páginas 0121–0126, 2020.

[30] C. Cecati, F. Ciancetta y P. Siano. A multilevel inverter for photovoltaic systems with fuzzy logic control. *IEEE Transactions on Industrial Electronics*, 57:4115–4125, 2010.

[31] C. Chakraborty y D. Chakraborty. A decision scheme based on OWA operator for an evaluation programme: An approximate reasoning approach. *Applied Soft Computing*, 5:45–53, 2004.

[32] C.-H. Cheng y J.-R. Chang. Mcdm aggregation model using situational Me-OWA and Me-Owga operators. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 14:421–443, 2006.

[33] N. Chinthamu y M. Karukuri. Data science and applications. *Journal of Data Science and Intelligent Systems*, 00(00):1–9, 2023.

[34] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1954.

[35] C.-H. Chou y J.-C. Teng. A fuzzy logic controller for traffic junction signals. *Information Sciences*, 143(1):73–97, 2002.

[36] A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936.

[37] A. Colmerauer, H. Kanoui, P. Roussel y R. Pasero. *Un Système de Communication Homme-Machine en Français*. Technical report, Groupe de Recherche en Intelligence Artificielle, Universidad de Aix-Marseille, 1973.

[38] A. Colmerauer y P. Roussel. The birth of Prolog. En *History of Programming Languages (HOPL) II*, páginas 331–367, Association for Computing Machinery, Nueva York, Nueva York, EE.UU., 1996.

[39] N. de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge Library Collection - Mathematics. Cambridge University Press, 2014.

[40] I. Copi y C. Cohen. *Introduction to Logic*. Maxwell Macmillan international editions, 1990.

[41] M. J. Cresswell y G. E. Hughes. *A New Introduction to Modal Logic*. Routledge, Nueva York, 1996.

[42] C. Damásio, J. Medina y M. Ojeda-Aciego. Termination of logic programs with imperfect information: applications and query procedure. *Journal of Applied Logic*, 5:435–458, 2007.

[43] C. V. Damásio y L. M. Pereira. Hybrid probabilistic logic programs as residuated logic programs. En *Logics in Artificial Intelligence, JELIA'00*, páginas 57–73. Lecture Notes in Artificial Intelligence 1919, 2000.

[44] C. V. Damásio y L. M. Pereira. Monotonic and residuated logic programs. En *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, páginas 748–759. Lecture Notes in Artificial Intelligence, 2143, 2001.

[45] M. S. Dattathreya, H. Singh y T. Meitzler. Detection and elimination of a potential fire in engine and battery compartments of hybrid electric vehicles. *Advances in Fuzzy Systems*, 2012(1):687652:1–687652:11, 2012.

[46] M. Davis y H. Putnam. A computing procedure for quantification theory. *Journal of the Association for the Computing Machinery*, 7(3):201–215, 1960.

[47] A. De Morgan. *On the Foundation of Algebra, No. II.* Transactions of the Cambridge Philosophical Society Offprint, 1842.

[48] A. De Morgan. *Formal Logic: Or, the Calculus of Inference, Necessary and Probable*. Taylor y Waldon, 1847.

[49] A. De Morgan. *On the Structure of the Syllogism: And on the Application of the Theory of Probabilities to Questions of Argument and Authority*. Transactions of the Cambridge Philosophical Society, 1847.

[50] M. Delgado, D. Sánchez y M. A. Vila. Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning*, 23(1):23–66, 2000.

[51] J. C. Díaz-Moreno, J. Medina y J. R. Portillo. Fuzzy logic programs as hypergraphs. Termination results. *Fuzzy Sets and Systems*, 445:22–42, 2022.

[52] R. Dixit y H. Singh. BDD, BNN, and FPGA on fuzzy techniques for rapid system analysis. *Advances in Fuzzy Systems*, 2012(1):290710:1–290710:9, 2012.

[53] S. Dotcenko, A. Vladyko y I. Letenko. A fuzzy logic-based information security management for software-defined networks. En *16th International Conference on Advanced Communication Technology*, páginas 167–171, 2014.

[54] M. Doumpos y C. Zopounidis. *Multicriteria decision aid classification methods*. Kluwer Academic Publishers, 2002.

[55] D. Dubois, L. Godo, R. Mántaras y H. Prade. Qualitative reasoning with imprecise probabilities. *Journal of Intelligent Information Systems*, 2:319–363, 1993.

[56] D. Dubois y H. Prade. A review of fuzzy set aggregation connectives. *Information Sciences*, 36(1):85–121, 1985.

[57] D. Dubois y H. Prade. Semantics of quotient operators in fuzzy relational databases. *Fuzzy Sets and Systems*, 78(1):89–93, 1996.

[58] A. Dumitriu. *History of Logic*. Abacus Press, Tunbridge Wells, 1977.

[59] A. Dvořák y M. Holčapek. L-fuzzy quantifiers of type $< 1 >$ determined by fuzzy measures. *Fuzzy Sets and Systems*, 160(23):3425–3452, 2009.

[60] P. Eklund y F. Klawonn. Neural fuzzy logic programming. *Transactions on Neural Networks*, 3(5):815–818, 1992.

[61] M. van Emden y R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Association for the Computing Machinery*, 23(4):733–742, 1976.

[62] H. B. Enderton. *A Mathematical introduction to logic*. Academic Press, San Diego, California, EE.UU., 1972.

[63] Y. Fitrian, M. Huda, A. Muhtar, A. Y. Arifin, N. Musa, K. S. Mat Teh, N. M. Sari y A. Maseleno. Application Design for Determining Suitable Cosmetics with the Facial Skin Type Using Fuzzy Logic Approach. *Journal of Computational and Theoretical Nanoscience*, 16(5):2153–2158, 2019.

[64] G. Frege. Begriffsschrift. En J. van Heijenoort, editor, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, páginas 1–83. Harvard University Press, Cambridge, Massachusetts, 1967.

[65] D. M. Gabbay, J. H. Siekmann y J. Woods. *Computational Logic, volumen 9*. North Holland & IFIP, 2014.

[66] M. Gelfond y V. Lifschitz. The stable model semantics for logic programming. En R. Kowalski, Bowen y Kenneth, editores, *Proceedings of International Logic Programming Conference and Symposium*, páginas 1070–1080. MIT Press, 1988.

[67] G. Gentzen. Untersuchungen über das logische schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1935.

[68] P. C. Gilmore. A proof method for quantification theory: Its justification and realization. *IBM Journal of Research and Development*, 4(1):28–35, 1960.

[69] V. D. Giudice, P. D. Paola y G. B. Cantisani. Valuation of real estate investments through fuzzy logic. *Buildings*, 7:26, 2017.

[70] I. Glöckner. *Fuzzy quantifiers in natural language:: semantics and computational models*. PhD thesis, Bielefeld University, 2004.

[71] I. Glöckner. *Fuzzy Quantifiers: A Computational Theory*, volumen 193. 2006.

[72] I. Glöckner. Evaluation of quantified propositions in generalized models of fuzzy quantification. *International Journal of Approximate Reasoning*, 37(2):93–126, 2004.

[73] K. Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme I. *Monatshefte für Mathematik*, 38(1):173–198, 1931.

[74] M. Grabisch. Fuzzy integral in multicriteria decision making. *Fuzzy Sets and Systems*, 69(3):279–298, 1995.

[75] C. Green. Application of theorem proving to problem solving. En *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, IJCAI'69, páginas 219–239, San Francisco, California, Estados Unidos, 1969. Morgan Kaufmann Publishers Inc.

[76] M. Gul, E. Celik, A. T. Gumus y A. F. Guneri. A fuzzy logic based PROMETHEE method for material selection problems. *Beni-Suef University Journal of Basic and Applied Sciences*, 7(1):68–79, 2018.

[77] S. Gupta y A. Gupta. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, 161:466–474, 2019.

[78] S. Haack. *Philosophy of Logics*. Cambridge University Press, 1978.

[79] G. Hägele y F. Pukelsheim. Llull's writings on electoral systems. *Studia Lulliana*, 41(97):3–38, 2001.

[80] P. Hajek. *The Metamathematics of Fuzzy Logic*. Kluwer Academic Publishers, 1998.

[81] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[82] P. J. Hayes. Computation and deduction. En *International Symposium on Mathematical Foundations of Computer Science*, páginas 105–117, Czechoslovak Academy of Sciences, 1973.

[83] L. Henkin. The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, 14:159 – 166, 1949.

[84] J. Herbrand. Investigations in proof theory. En J. van Heijenoort, editor, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, páginas 525-581. Harvard University Press, Cambridge, Massachusetts, 1967.

[85] E. Herrera-Viedma y E. Peis. Evaluating the informative quality of documents in SGML format using fuzzy linguistic techniques based on computing with words. *Information Processing & Management*, 39:233–249, 2003.

[86] A. Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preußischen Akademie der Wissenschaften, Physikalisch-Mathematische Klasse*, 42–56, 1930.

[87] D. Hilbert y W. Ackermann. *Grundzüge der Theoretischen Logik*. Springer Berlín, Heidelberg, 1928.

[88] D. Hilbert, M. W. Newsom, F. E. Browder, D. A. Martin, G. Kreisel y M. Davis. Mathematical problems. Lecture delivered before the International Congress of Mathematicians at Paris in 1900. *Journal of Symbolic Logic*, 44(1):116–119, 1979.

[89] K. Hirota. Industrial applications and education on fuzzy systems in Japan. *SBA Controle & Automação*, 4(3):155–161, 1994.

[90] A. Hodges y D. Hofstadter. *Alan Turing: The Enigma*. Walker & Company, 2000.

[91] L. Holmblad y J.-J. Østergaard. The FLS application of fuzzy logic. *Fuzzy Sets and Systems*, 70(2):135–146, 1995.

[92] M. Holčapek. Monadic L-fuzzy quantifiers of the type $< 1^n, 1 >$. *Fuzzy Sets and Systems*, 159(14):1811–1835, 2008.

[93] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14 – 21, 1951.

[94] J. Jaffar y J.-L. Lassez. Constraint logic programming. En *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '87, páginas 111–119, Nueva York, Nueva York, Estados Unidos, 1987.

[95] S. Jaśkowski. Propositional calculus for contradictory deductive systems. *Studia Logica*, 24(1):143–160, 1969.

[96] Z. C. Johanyák. Fuzzy logic based network intrusion detection systems. En *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, páginas 15–16, 2020.

[97] P. Julián-Iranzo, J. Medina y M. Ojeda-Aciego. On reductants in the framework of multi-adjoint logic programming. *Fuzzy Sets and Systems*, 317:27 – 43, 2017.

[98] P. Julián-Iranzo y C. Rubio-Manzano. Bousi~Prolog - A fuzzy logic programming language for modeling vague knowledge and approximate reasoning. En J. Filipe y J. Kacprzyk, editores, *ICFC-ICNC*

2010 - Proceedings of the International Conference on Fuzzy Computation and International Conference on Neural Computation, [parts of the International Joint Conference on Computational Intelligence IJCCI 2010], Valencia, España, Octubre 24-26, 2010, páginas 93–98. SciTePress, 2010.

[99] P. Julián-Iranzo, G. Moreno y J. A. Riaza. The fuzzy logic programming language FASILL: Design and implementation. *International Journal of Approximate Reasoning*, 125:139–168, 2020.

[100] P. Julián-Iranzo y F. Sáenz-Pérez. Bousi-Prolog: Design and implementation of a proximity-based fuzzy logic programming language. *Expert Systems with Applications*, 213:118858, 2023.

[101] J. Kacprzyk. Intelligent data analysis via linguistic data summaries: A fuzzy logic approach. En R. Decker y W. Gaul, editores, *Classification and Information Processing at the Turn of the Millennium*, páginas 153–161. Springer Berlín Heidelberg, 2000.

[102] J. Kacprzyk y S. Zadrożny. Computing with words in intelligent database querying: standalone and internet-based applications. *Information Sciences*, 134(1):71–109, 2001.

[103] J. Kacprzyk, S. Zadrożny, M. Fedrizzi y H. Nurmi. *On Group Decision Making, Consensus Reaching, Voting and Voting Paradoxes under Fuzzy Preferences and a Fuzzy Majority: A Survey and some Perspectives*, páginas 263–295. Springer Berlín Heidelberg, 2008.

[104] C. Kahraman, S. Çevik y B. Öztayşi. Fuzzy multicriteria decision-making: A literature review. *International Journal of Computational Intelligence Systems*, 8:637–666, 2015.

[105] A. C. Kakas, R. A. Kowalski y F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 12 1992.

[106] D. M. Kan. Adjoint functors. *Transactions of the American Mathematical Society*, 87(2):294–329, 1958.

[107] R. Karthik y S. Ganapathy. A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce. *Applied Soft Computing*, 108:107396, 2021.

[108] H. J. Keisler. Logic with the quantifier "there exist uncountably many". *Annals of Mathematical Logic*, 1(1):1–93, 1970.

[109] G. Klir y B. Yuan. *Fuzzy sets and fuzzy logic*, volumen 4. Prentice Hall Nueva Jersey, 1995.

[110] W. C. Kneale y M. Kneale. *The Development of Logic*. Clarendon Press, Oxford, England, 1962.

[111] R. Kowalski. Predicate logic as programming language. En *Proceedings of IFIP'74*, páginas 569–574. North Holland Publishing Company, 1974.

[112] R. Kowalski. *Logic for Problem Solving*. Elsevier North Holland, 1979.

[113] R. Kowalski y D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2(3):227–260, 1971.

[114] R. A. Kowalski. Algorithm = logic + control. *Communications of the Association for Computing Machinery*, 22:424–436, 1979.

[115] P. Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32(3):186–195, 1966.

[116] Y. Liu y E. E. Kerre. An overview of fuzzy quantifiers. (I). interpretations. *Fuzzy Sets and Systems*, 95(1):1–21, 1998.

[117] I. Lizasoain y C. Moreno. OWA operators defined on complete lattices. *Fuzzy Sets and Systems*, 224:36 – 52, 2013.

[118] J. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.

[119] M. Mahout, R. P. Carlson, L. Simon y S. Peres. Logic programming-based minimal cut sets reveal consortium-level therapeutic targets for chronic wound infections. *npj Systems Biology and Applications*, 10(1):34, 2024.

[120] E. Mamdani y S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[121] H. Mathiassen y D. Ortiz-Arroyo. Automatic categorization of patent applications using classifier combinations. En E. Corchado, H. Yin, V. Botti y C. Fyfe, editores, *Intelligent Data Engineering and Automated Learning – IDEAL 2006*, páginas 1039–1047. Springer Berlín Heidelberg, 2006.

[122] J. Medina y J. A. Torné-Zambrano. Computational reaching of quantified consequences from imperfect initial data. *Mathematical Methods in the Applied Sciences*, 2024.

[123] J. Medina y J. Moreno-García. Modeling photovoltaic facilities via fuzzy sets and linguistic petri nets. *International Journal of Information Technology & Decision Making*, 2024. Pendiente de publicación.

[124] J. Medina, M. Ojeda-Aciego y P. Vojtáš. Multi-adjoint logic programming with continuous semantics. En *Logic Programming and Non-Monotonic Reasoning, LPNMR'01, Lecture Notes in Artificial Intelligence 2173*, páginas 351–364. Springer, 2001.

[125] J. Medina, M. Ojeda-Aciego y P. Vojtáš. A procedural semantics for multi-adjoint logic programming. En *Progress in Artificial Intelligence, EPIA'01, Lecture Notes in Artificial Intelligence 2258*, páginas 290–297. Springer, 2001.

[126] J. Medina, K. Pakhomova y E. Ramírez-Poussa. Recommendation solution for a locate-based social network via formal concept analysis. *Studies in Computational Intelligence*, 796:131–138, 2019.

[127] J. Medina y J. A. Torné-Zambrano. Preference in Multi-adjoint Logic Programming Based on Ordered Adjoint Pairs, En M. Cornejo, L.T. Kóczy, J. Medina y E. Ramírez-Poussa, editores, *Computational Intelligence and Mathematics for Tackling Complex Problems 5. Studies in Computational Intelligence*, volumen 1127, páginas 93–99. Springer Nature Switzerland, Cham, 2024.

[128] J. Medina y J. A. Torné-Zambrano. Operational semantics on arbitrary first-order fuzzy logic programming. Enviado.

[129] J. Medina y J. A. Torné-Zambrano. Immediate consequences operator on generalized quantifiers. *Fuzzy Sets and Systems*, 456:72–91, 2022.

[130] J. Medina y J. A. Torné-Zambrano. Immediate consequences operator on through ordered weighted average operators. *Iranian Journal of Fuzzy Sets*, 20(5):1–18, 2023.

[131] J. Medina y J. A. Torné-Zambrano. Computational technique for obtaining consequences in multi-adjoint logic programming with generalized quantifiers. *Applied and Computational Mathematics*, 23(4):520–540, 2024.

[132] K. Menger. Statistical metrics. *Proceedings of the National Academy of Sciences of the United States of America*, 28(12):535–537, 1942.

[133] M. R. H. Mohd Adnan, A. Sarkheyli-Hägele, A. Zain y H. Haron. Fuzzy logic for modeling machining process: a review. *Artificial Intelligence Review*, 43, 2013.

[134] P. J. Morcillo, G. Moreno, J. Penabad y C. Vázquez. A practical management of fuzzy truth-degrees using floper. En M. Dean, J. Hall, A. Rotolo y S. Tabet, editores, *Semantic Web Rules*, páginas 20–34. Springer Berlín Heidelberg, 2010.

[135] G. Moreno, J. Pérez y J. A. Riaza. Fuzzy logic programming for tuning neural networks. En P. Fodor, M. Montali, D. Calvanese y D. Roman, editores, *Rules and Reasoning*, páginas 190–197. Springer International Publishing, Cham, 2019.

[136] A. Mostowski. On a generalization of quantifiers. *Fundamenta Mathematicae*, 44(2):12–36, 1957.

[137] A. Mudassir, S. Akhtar, H. Kamel y N. Javaid. A survey on fuzzy logic applications in wireless and mobile communication for lte networks. En *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, páginas 76–82, 2016.

[138] S. Muggleton y L. de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994.

[139] M. S. Munir, I. S. Bajwa y S. M. Cheema. An intelligent and secure smart watering system using fuzzy logic and blockchain. *Computers & Electrical Engineering*, 77:109–119, 2019.

[140] B. M. Nair y J. Cai. A fuzzy logic controller for isolated signalized intersection with traffic abnormality considered. En *2007 IEEE Intelligent Vehicles Symposium*, páginas 1229-1233. IEEE, 2007.

[141] T. Nguyen, A. Khosravi, D. Creighton y S. Nahavandi. Classification of healthcare data using genetic fuzzy logic system and wavelets. *Expert Systems with Applications*, 42(4):2184–2197, 2015.

[142] V. Novák. A formal theory of intermediate quantifiers. *Fuzzy Sets and Systems*, 159(10):1229–1246, 2008.

[143] M. O'Hagan. Aggregating template or rule antecedents in real-time expert systems with fuzzy set logic. En *Twenty-Second Asilomar Conference on Signals, Systems and Computers*, volumen 2, páginas 681–689. IEEE, 1988.

[144] B. Ojokoh, O. Omisore, O. Samuel y O. Ogunniyi. A fuzzy logic based personalized recommender system. *International Journal of Computer Science, Information Technology, Security*, 2:2249–9555, 2012.

[145] E. Pap. Aggregation Operators in Engineering Design. En *Studies in Fuzziness and Soft Computing*, volumen 97, páginas 195–223. Springer, 2002.

[146] J. Pavelka. On fuzzy logic I: many-valued rules of inference. *Mathematical Logic Quarterly*, 25(3-6):45–52, 1979.

[147] J. Pavelka. On fuzzy logic II: enriched residuated lattices and semantics of propositional calculi. *Mathematical Logic Quarterly*, 25(7-12):119–134, 1979.

[148] J. Pavelka. On fuzzy logic III. semantical completeness of some many-valued propositional calculi. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 25(25-29):447–464, 1979.

[149] G. Peano. *Arithmetices principia: nova methodo*. Fratres Bocca, 1889.

[150] G. Peano. *Notations de logique mathématique*. Guadagnini, 1894.

[151] G. Peano. Formulaire de mathématiques. *Revue de Métaphysique et de Morale*, 7(1):5–6, 1899.

[152] C. S. Peirce. On the algebra of logic: A contribution to the philosophy of notation. *American Journal of Mathematics*, 7:180.

[153] S. Peters y D. Westerståhl. *Quantifiers in Language and Logic*. Clarendon Press, Oxford, Inglaterra, 2006.

[154] O. Pivert y P. Bosc. *Fuzzy Preference Queries to Relational Databases*. Imperial College Press, 2012.

[155] D. Prawitz. An improved proof procedure. *Theoria*, 26:102 – 139, 1960.

[156] A. N. Prior. *Time and Modality*. Greenwood Press, Westport, Connecticut, 1955.

[157] F. Provost y T. Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media, Inc., 1st edition, 2013.

[158] A. L. Ralescu. A note on rule representation in expert systems. *Information Sciences*, 38(2):193–203, 1986.

[159] A. L. Ralescu, D. A. Ralescu y K. Hirota. Evaluation of fuzzy quantified expressions. En A. L. Ralescu y J. G. Shanahan, editors, *Fuzzy Logic in Artificial Intelligence*, páginas 234–244. Springer Berlín Heidelberg, 1999.

[160] D. Rasmussen y R. R. Yager. A fuzzy SQL summary language for data discovery. *Intelligent Data Analysis*, 1(1-4):49–58, 1997.

[161] H. Rezk, M. Aly, M. Al-Dhaifallah y M. Shoyama. Design and hardware implementation of new adaptive fuzzy logic-based MPPT control method for photovoltaic applications. *IEEE Access*, 7:106427–106438, 2019.

[162] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for the Computing Machinery*, 12(1):23–41, 1965.

[163] F. P. Romero, P. Julián-Iranzo, A. Soto, M. Ferreira-Satler y J. Gallardo-Casero. Classifying unlabeled short texts using a fuzzy declarative approach. *Language Resources and Evaluation*, 47(1):151–178, 2013.

[164] C. Rubio-Manzano y P. J. Iranzo. Incorporation of abstraction capability in a logic-based framework by using proximity relations. *Journal of Intelligent & Fuzzy Systems*, 29:1671–1683, 2015.

[165] M. D. Ruiz, D. Sánchez y M. Delgado. On the relation between fuzzy and generalized quantifiers. *Fuzzy Sets and Systems*, 294:125–135, 2016.

[166] B. Russell. *Principles of Mathematics*. Routledge, Nueva York, 1903.

[167] B. Saha y D. Srivastava. Data quality: The other face of big data. En *2014 IEEE 30th International Conference on Data Engineering*, páginas 1294–1297, 2014.

[168] S. Sandri, D. Dubois y H. Kalfsbeek. Elicitation, assessment, and pooling of expert judgments using possibility theory. *Fuzzy Systems, IEEE Transactions on*, 3:313 – 335, 1995.

[169] A. Sarkar, G. Sahoo y U. C. Sahoo. Application of fuzzy logic in transport planning. *International Journal of Soft Computing*, 3:1–21, 2012.

[170] T. Sato. A statistical learning method for logic programs with distribution semantics. En *International Conference on Logic Programming*, The MIT Press, 1995.

[171] B. Schweitzer y A. Sklar. Associative functions and statistical triangle inequalities. *Publicationes Mathematicae Debrecen*, 2022.

[172] D. Scott. Outline of a mathematical theory of computation. *Kibernetticheskij Sbornik. Novaya Seriya*, 14, 1970.

[173] J. Serrano-Guerrero, F. P. Romero y J. A. Olivas. Fuzzy logic applied to opinion mining: A review. *Knowledge-Based Systems*, 222:107018, 2021.

[174] J. Sgro. Completeness theorems for topological models. *Annals of Mathematical Logic*, 11(2):173–193, 1977.

[175] E. Shapiro. *A subset of Concurrent Prolog and its interpreter*, páginas 27–83. MIT Press, EE.UU., 1988.

[176] A. Sharaff y P. G. Sinha. *Data Science and Its Applications*. Chapman and Hall, CRC Press, 2021.

[177] M. Shlibek. Fuzzy logic based control system for intelligent washing machines. *European Journal of Engineering Science and Technology*, 2019.

[178] M. Sicilia, E. Barriocanal y S. Alonso. Empirical assessment of a collaborative filtering algorithm based on OWA operators. *International Journal of Intelligent Systems*, 23:1251–1263, 2008.

[179] N. Silver. *The signal and the noise: why so many predictions fail–but some don't*. Penguin, Nueva York, 2012.

[180] W. Slany. Scheduling as a fuzzy multiple criteria optimization problem. *Fuzzy Sets and Systems*, 78(2):197–222, 1996.

[181] E. Solares, V. De-León-Gómez, F. G. Salas y R. Díaz. A comprehensive decision support system for stock investment decisions. *Expert Systems with Applications*, 210:118485, 2022.

[182] M. Spies. Inferenzen bei Ungewißheit in Expertensystemen. En D. Metzing, editor, *GWAI-89 13th German Workshop on Artificial Intelligence*, páginas 428–432. Springer Berlín Heidelberg, 1989.

[183] A. Srinivasan, A. Baskar, T. Dash y D. Shah. Composition of relational features with an application to explaining black-box predictors. *Machine Learning*, 113(3):1091–1132, 2024.

[184] L. Sterling y E. Shapiro. *The art of Prolog (2nd ed.): advanced programming techniques*. MIT Press, Cambridge, Massachusetts, Estados Unidos, 1994.

[185] V. S. Subrahmanian. On the semantics of quantitative logic programs. En *International Conference on Spoken Language Processing*, páginas 173–182. IEEE, 1987.

[186] L. Suganthi, S. Iniyan y A. A. Samuel. Applications of fuzzy logic in renewable energy systems–a review. *Renewable and sustainable energy reviews*, 48:585–607, 2015.

[187] M. Sugeno. Theory of fuzzy integrals and its applications. *Doctoral thesis tokyo institute of technology*, 1974.

[188] A. Tarski. *Introduction to Logic and to the Methodology of Deductive Sciences*. Dover Publications, Nueva York, 1946.

[189] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[190] H. Thöne, U. Güntzer y W. Kiessling. Towards precision of probabilistic bounds propagation. En *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, páginas 315-322, 1992.

[191] L. Tomás, C. Vázquez Pérez-Íñigo, J. Tordsson y G. Moreno. Reducing noisy-neighbor impact with a fuzzy affinity-aware scheduler.

En *2015 International Conference on Cloud and Autonomic Computing*, páginas 33–44, 2015.

[192] V. Torra. OWA operators in data modeling and reidentification. *IEEE Transactions on Fuzzy Systems*, 12(5):652–660, 2004.

[193] V. Torra. *The WOWA Operator: A Review*, páginas 17–28. Springer Berlín Heidelberg, Berlín, Heidelberg, 2011.

[194] A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265, 1936.

[195] J. van Benthem. Questions about quantifiers. *Journal of Symbolic Logic*, 49(2):443–466, 1984.

[196] P. Vojtáš. Fuzzy logic programming. *Fuzzy sets and systems*, 124(3):361–370, 2001.

[197] M. Štěpnička y M. Holčapek. Fuzzy relational compositions based on generalized quantifiers. *Communications in Computer and Information Science*, 443:224–233, 2014.

[198] H. Wang. Toward mechanical mathematics. *IBM Journal of Research and Development*, 4(1):2–22, 1960.

[199] R. Y. Wang y D. M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12:5–33, 1996.

[200] A. N. Whitehead y B. Russell. *Principia Mathematica*. Cambridge University Press, 1925–1927.

[201] T. Williamson. Vagueness. *British Journal for the Philosophy of Science*, 46(4):589–601, 1994.

[202] R. Yager. OWA neurons: a new class of fuzzy neurons. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volumen 1, páginas 226–231, 1992.

[203] R. Yager. Toward a language for specifying summarizing statistics. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 33:177–87, 2003.

[204] R. Yager. Time series smoothing and OWA aggregation. *IEEE Transactions on Fuzzy Systems*, 16:994 – 1007, 2008.

[205] R. Yager, D. Filev y T. Sadeghi. Analysis of flexible structured fuzzy logic controllers. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(7):1035–1043, 1994.

[206] R. R. Yager. A new approach to the summarization of data. *Information Sciences*, 28(1):69–86, 1982.

[207] R. R. Yager. Reasoning with fuzzy quantified statements: Part I. *Kybernetes*, 14(4):233–240, 1985.

[208] R. R. Yager. Reasoning with fuzzy quantified statements: Part II. *Kybernetes*, 15(2):111–120, 1986.

[209] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[210] R. R. Yager. On linguistic summaries of data. *Knowledge Discovery in Databases*, páginas 347–366, 1991.

[211] R. R. Yager. Families of OWA operators. *Fuzzy Sets and Systems*, 59(2):125 – 148, 1993.

[212] R. R. Yager. Implementing quotient operators in fuzzy data bases using OWA operators. En P. Bosc y J. Kacprzyk, editors, *Fuzziness in Database Management Systems*, páginas 186–206, Heidelberg, 1995. Physica-Verlag HD.

[213] R. R. Yager y D. P. Filev. *Essentials of fuzzy modeling and control*. Wiley-Interscience, Estados Unidos, 1994.

[214] R. R. Yager y J. Kacprzyk. *The ordered weighted averaging operators: Theory and applications.* Springer Publishing Company, Inc., 1997.

[215] L. Zadeh. Fuzzy algorithms. *Information and Control*, 12(2):94–102, 1968.

[216] L. Zadeh. PRUF-a meaning representation language for natural languages. *International Journal of Man-Machine Studies*, 10(4):395–460, 1978.

[217] L. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11(1):199–227, 1983.

[218] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[219] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:28–44, 1973.

[220] L. A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computers & Mathematics with Applications*, 9(1):149 – 184, 1983.

[221] L. A. Zadeh. *A Theory of Commonsense Knowledge*, páginas 257–295. Springer Netherlands, Dordrecht, 1984.

[222] L. A. Zadeh. Syllogistic reasoning in fuzzy logic and its application to usuality and reasoning with dispositions. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(6):754–763, 1985.

[223] S. Zadrożny y J. Kacprzyk. On tuning OWA operators in a flexible querying interface. En H. L. Larsen, G. Pasi, D. Ortiz-Arroyo, T. Andreasen y H. Christiansen, editores, *Flexible Query Answering Systems*, páginas 97–108. Springer, Berlín, Heidelberg, 2006.