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] $(L, \preceq, \leftarrow_1, \&_1, \dots, \leftarrow_n, \&_n)$, a negation operator $\neg$, and other operators, such as conjunctors $\wedge_1, \dots, \wedge_k$, disjunctors $\vee_1, \dots, \vee_l$ and general aggregators $@_1, \dots, @_h$. 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 $@[B_1, \dots, B_s, \neg B_{s+1}, \dots, \neg B_r]$ built from propositional symbols $B_1, \dots, B_r$ ($r \geq 0$, $B_i \neq B_j$, for $i \neq j$) by the use of conjunctors $\&_1, \dots, \&_n$ and $\wedge_1, \dots, \wedge_k$, disjunctors $\vee_1, \dots, \vee_l$, aggregators $@_1, \dots, @_m$ 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, $(\&_P, \leftarrow_P)$, $(\&_G, \leftarrow_G)$, $(\&_Ł, \leftarrow_Ł)$ 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 $\langle p \leftarrow_i @[p_1, \dots, p_m, \neg p_{m+1}, \dots, \neg p_n]\,; \vartheta\rangle$, the following statements hold:

- $||p_i|| \leq ||p||$ for each $i \in \{1, \dots, m\}$.
- $||p_i|| < ||p||$ for each $i \in \{m+1, \dots, n\}$.

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 $\{M_1^{||\cdot||}, M_2^{||\cdot||}, \dots\}$ 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

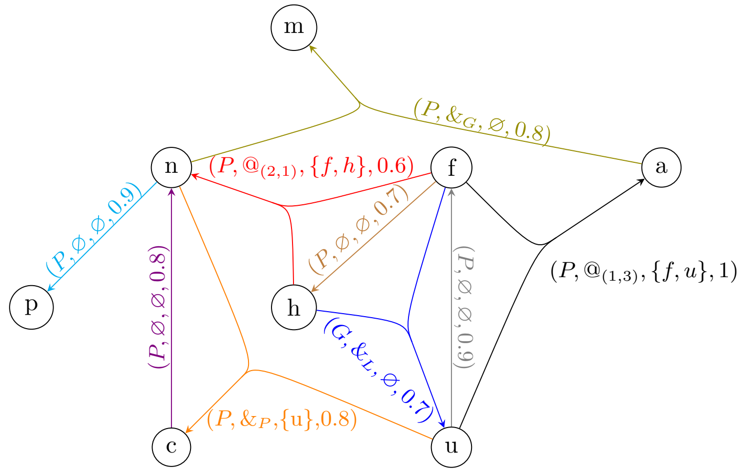

Fig. 1. Edge-labeled B-graph associated with the program given in Example 1. The facts of the program (rules 8, 9 and 10 of Example 1) are not shown by simplicity.

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: $\{\mathrm{f, h, u}\}$, $\{\mathrm{c, n}\}$, and $\{\mathrm{a}\}$.

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.

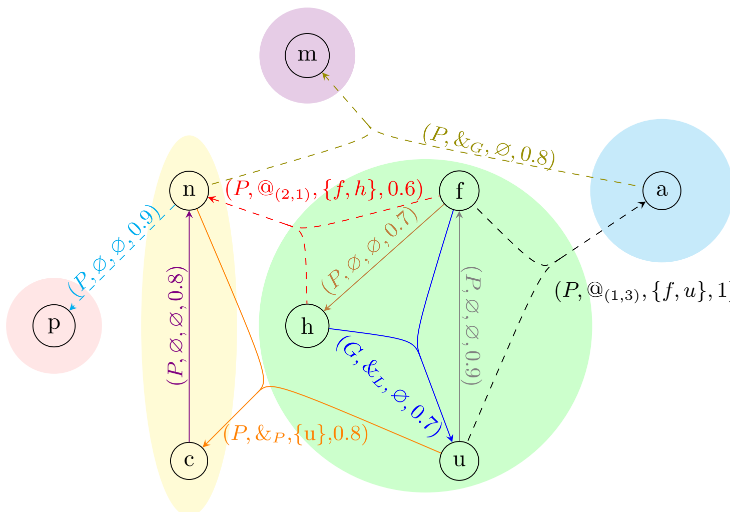

Fig. 2. SPC-components of the edge-labeled B-graph associated with the program given in Example 1. The d-hyperarcs are shown with dashed lines.

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.

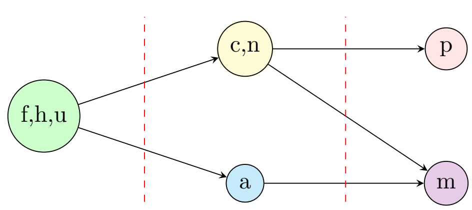

Fig. 3. p-condensation digraph of the hypergraph in Figure 1. Vertices are labelled with the vertices of the corresponding SPC of the hypergraph.

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.