Research Summary: Explainable Artificial Intelligence in Answer Set Programming and Machine Learning

Ly Ly Trieu[1,*]

[1]Department of Computer Science, New Mexico State University, Las Cruces, NM 88003, USA

Abstract

Explanations for Answer Set Programming and Machine Learning fall within the scope of *explainable Artificial Intelligence* (xAI), which aims to make intelligent systems more transparent. This research not only enhances the understanding of the provided solutions or questions, but also strengthens user trust, supports better decision-making. In this work, we present a research summary on xAI, focusing on how the state-of-the-art approaches in Answer Set Programming and Machine Learning generate the explanations or justifications for the solutions or questions. The summary integrates both existing literature and our own research contributions, outlining the progress made in the xAI field.

Keywords

Answer Set Programming, eXplainable Artificial Intelligence, Knowledge Representation and Reasoning, Deep Neural Networks

1. Introduction and Problem Description

Intelligent systems have recently demonstrated a strong capability to successfully tackle complex problems [1, 2] and have been applied in various domains such as health care [3, 4, 5] and transportation [6]. However, as a system's performance reaches a higher level, its transparency diminishes [1, 7, 8, 9]. DARPA [7] highlights the trade-off between performance metrics, such as accuracy and explainability. The lack of transparency raises mistrust among users, especially in critical domains such as healthcare, human life, law, privacy, and finance [1, 10, 11]. As a result, the interest in explainable artificial intelligence (xAI) has grown significantly in recent years. xAI systems aim to provide explanations for their results so that the users can understand why they receive the results such as predictions and recommendations; thus, they can decide whether they should trust the results.

The recognition of xAI has been amplified by a critical legal provision known as the *right to an explanation* law, instituted by The European Parliament [12, 13, 14, 9]. The General Data Protection Regulation (GDPR) became law in May 2018. Article 22 in GDPR states that anyone has the right to reject a "decision based solely on automated processing" that "signicantly affects" this person [12]. Implicit in the GDPR is that, in the future, the automated decision must be explicable in human-understandable manner for it to be acceptable [15, 12]. Consequently, the implementation for integrating applications equipped with xAI capabilities has become urgent, presenting a substantial scientific challenge.

As xAI becomes increasingly important, this research summary presents two different methodologies in this field, data-centric approach which is deep neural networks (DNNs), and rule-based approach which is answer set programming (ASP). In ASP, identifying the reason for the presence or absence of a given atom in an answer set can be a challenging task due to the many possible interactions with large knowledge/logic program. Explainability for deep neural networks has been a topic of intensive research due to the meteoric rise in prominence of deep neural networks and "black-box" nature of their decision-making.

ICLP DC 2025: 21st Doctoral Consortium on Logic Programming, September 2025, Rende, Italy.

*Corresponding author.

lytrieu@nmsu.edu (L. L. Trieu)

https://www.cs.nmsu.edu/~ltrieu/ (L. L. Trieu)

0000-0003-1482-9453 (L. L. Trieu)

The organization of this work is as follows. Chapter 2 introduces the background of ASP and DNN. The existing literature is presented in Chapter 3. Chapter 4 outlines the research goal. Chapter 5 presents the current status and results. Chapter 6 discusses future work and conclusions.

2. Background

2.1. Answer Set Programming

Answer Set Programming (ASP) [16, 17] is a declarative programming paradigm based on logic programming under the answer set semantics. A logic program P is a set of rules of the form "$c \leftarrow a_1,\ldots,a_m,not\ b_1,\ldots,not\ b_n$" where c, a_i's, and b_j's are atoms of a first order language. The semantics of logic programs is often defined over the ground instantiation of P. Therefore, to simplify the presentation we will assume that a program P is propositional program. *not* represents (default) negation. Intuitively, a rule states that if all a_i's are believed to be true and none of the b_j's is believed to be true then c will be true. For a rule r, r^+ and r^- denote the sets $\{a_1,\ldots,a_m\}$ and $\{b_1,\ldots,b_n\}$, respectively. We write $head(r)$ and $body(r)$ to denote c and the right side of a rule r. Both the head and the body of a rule r can be empty; when the body is empty, the rule is called a fact; when the head is empty, it is a constraint. We use H to denote the Herbrand base of a logic program P, which is the set of all ground atoms in P.

Let P be a program. An interpretation I of P is a subset of H. I satisfies an atom a ($I \models a$) if $a \in I$. The body of a rule r is satisfied by I if $r^+ \subseteq I$ and $r^- \cap I = \emptyset$. A rule r is satisfied by I if $I \models head(r)$ or $I \not\models body(r)$. I is a model of P if it satisfies all rules in P.

For an interpretation I and a program P, the *reduct* of P w.r.t. I (denoted by P^I) is the program obtained from P by deleting *(i)* each rule r such that $r^- \cap I \neq \emptyset$, and *(ii)* all elements of the form *not a* in the bodies of the remaining rules. Given an interpretation I, observe that the program P^I is a program with no occurrences of *not a*. An interpretation I is an *answer set* of P if I is the least model (w.r.t. $\subseteq$) of P^I [18]. Answer sets of logic programs can be computed using efficient and scalable answer set solvers, such as *clingo* [19].

2.2. Deep Neural Network

A neural network consists of an input layer i, an output layer o, and multiple hidden layers $h_1,\ldots,h_j$, $j \geq 1$, between them ([20]). Each layer consists of a set of nodes, each node is associated with a bias and an activation function. Node j at layer $l-1$ provides an input with weight w_{ij}^l to node i at layer l. The matrices representing the weights and biases of the network are denoted by $\mathbf{w}$ and $\mathbf{b}$, respectively. The input layer i receives raw features from data, while the output layer o produces the network's prediction.

The number of nodes in each layer depends on the application and the designer of the network. Usually, the number of nodes in the input and output layer corresponds to the number of features in the input data and the number of classes of the classification task, respectively.

Given a training set[1] $\{(\mathbf{x_i}, y_i) \mid i = 1,...,N\}$, the goal of training a network is to learn $\mathbf{w}$ to minimize the error function $E(\mathbf{w},\mathbf{b}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ where $\hat{y}_i$ represents the output value of the network given the input $\mathbf{x_i}$. a_i^l is the activation value at node i at layer l, whose activation function is f, is computed by $a_i^l = f(\sum_j w_{ij}^l a_j^{l-1} + b_i^l)$ (See, e.g., ([21]) for more details). Different activation functions can be used (e.g., *Tanh*, *Sigmoid*, or *ReLU*).

3. Overview of the Existing Literature

3.1. Answer Set Programming

Our work focuses on xAI for ASP, which in turn can be applied to debug by identifying a set of rules that justifies the derivation of a given atom. For example, if an atom α is supposed to be false in all

[1] For simplicity of the representation, we assume a network with one node at the output layer. Multiple output nodes are formalized similarly.

answer sets of a program Π but appears in some answer set A, an explanation graph of α could help to understand which rules are behaving anomalously. Therefore, in this section, we consider some debugging tools for ASP (e.g. `spock` and `DWASP`), as well as state-of-the-art XAI systems for ASP (e.g. `xclingo`, `xclingo 2.0`, `s(CASP)` and off-line justifications).

Several xAI systems have been developed to provide explanations for ASP. Among them, the following are particularly notable. `xclingo` [22] generates derivation trees for an atom by adding `trace_rule` and `trace` annotations to the input program. These annotations are compiled into theory atoms and auxiliary predicates, and the resulting explanations are derived by decoding the answer sets of the modified program. `xclingo 2.0`, the latest release of `xclingo` [23, 24], introduces new annotations: `mute` for atoms and `mute_body` for rules. The annotations serve to prune the edges and exclude nodes explanations, thus aiding in information filtering. `s(CASP)` [25] leverages top-down Prolog computation to generate a justification tree in natural language for Constraint Answer Set programs. Additionally, explanation graphs can be given in terms of off-line justifications [26, 27], possibly containing cycles among false atoms [26].

Some well-known debugging tools are as follows. `spock` [28] makes use of tagging techniques [29] to translate the input program into a new program whose answer sets can be used to debug the original program. The information reported to the user includes rules whose body is true (applicable rules), rules whose body is false (blocked rules), and abnormality tags associated with completion and loop formulas. `DWASP` [30] is aimed at identifying a set of rules that are responsible for the absence of an expected answer set. It combines the grounder `gringo` [31] and an extension of the ASP solver `WASP` [32], and introduces the `gringo-wrapper` to "disable" some grounding simplifications.

Current approaches face several limitations: for example, `xclingo` and `xclingo 2.0` cannot include negative literals in their explanations; `s(CASP)` generates different justifications when the order of atoms in rules or the order of rules in the program changed; and some methods (e.g. off-line justification) lack support for extended language constructs such as aggregates.

3.2. Deep Neural Network

The study in [13] classifies xAI methods into *local* and *global* explanations. Local explanation refers to the ability of explaining a specific prediction, while global one pertains to the overall model, covering all predictions. A well-known method for local explanations is Local Interpretable Model-Agnostic Explanations (LIME) ([33]). In contrast, SHapley Additive exPlanation, `SHAP` ([34]), is a model-agnostic, statistically-based method that uses Shapley values to explain feature importance globally.

Rule extraction methods have also been developed for understanding DNNs. Some methods focus on generating counterfactuals which help identifying features that need to be changed to produce a different output ([35]). These methods allow users to understand how small changes in input can impact the model's decision. Some other approaches construct IF-THEN rules from the trained DNN model which are then used to explain the dependence between input feature values and outputs. This provides users with the understanding of the model behavior across all inputs. The work in [36] is a well-known method that proposes a procedure for inducing logic-based theory for a given neural model (referred to as the main network). The method has to utilize multiple mapping networks that are trained to map the activation values of the main network's output to human-defined concepts. The primary cost of this approach lies in the need to label data for the mapping network, which usually requires domain expert knowledge. Following a taxonomy proposed by [10] and [37], rule extraction approaches can be divided into three categories: decompositional, pedagogical and eclectic. Decompositional methods derive rules based on the structure of the trained neural network and take into consideration hidden layers during extraction (e.g. `DeepRED` [38] and `ECLAIRE`[39]). On the other hand, pedagogical methods disregard the hierarchical structure in the extraction process (e.g. `HYPINV` [40] and `RxREN` [41]). Eclectic methods like `Modified RX+CGA`[42] integrate both decompositional and pedagogical approaches.

It is worth noticing that some feature/rules extraction methods directly from data ([43, 44] such as linear discriminant analysis (LDA) and decision tree could also be used to identify the importance of features and interpretable results. However, these methods do not explain the behavior of the given `DNN`

and have difficulty dealing with irrelevant features.

4. Goal of the Research

My research is in the field of Explainable Artificial Intelligence, which focuses on generating explanations or justifications for the outputs of intelligent systems. The goal of my work is to develop generation explanation systems for AI, which is achieved by exploring two methodologies:

1. The data-centric approach using deep neural networks (DNNs): DNNs, which benefit from extracting features from large datasets, are widely used in applications like speech recognition and classification. This work presents post-hoc explanation method for DNNs that analyze trained models to uncover their decision-making processes.
2. The rule-based approach using answer set programming (ASP): ASP is particularly attractive due to its non-monotonic reasoning, elaboration tolerance, and scalable solvers. The explanation of ASP solutions focuses on understanding why a particular answer set is a solution to a given problem, or why an atom (a logical consequence) is included or excluded from an answer set in a given program.

5. Current Status of the Research and Results

5.1. XAI for Answer Set Programming

Generating faithful explanations for True and False atoms. One of my first works, exp(ASP) [45], which based on [26], generates explanation graphs that explain why a is true (or false) given the (grounded/ungrounded) logic program P and its answer set A. These graphs include information about the applicable rules and atoms that contribute to the truth value of a. In exp(ASP), the program P is preprocessed to preserve facts and as many ground rules as possible by using the –text and –keep-facts options, along with converting facts into the external statements. The aspif representation of P is then obtained and processed, together with A, to generate explanation graphs through the following steps: (i) determining the set of assumptions U with respect to A; (ii) generating explanation graphs for a using P, A, and U following its definition adapted from [26]). exp(ASP) has been applied to an ASP program that analyzes the most severe nuclear accident in U.S. history–the Three Mile Island Unit 2 (TMI-2) accident scenario–to generate explanations that help answer questions such as "Why did an event occur?" or "What should be done?" [46]. Later, I developed exp(ASP^{c}) [47] as an enhancement of exp(ASP), improving its ability to support choice rules and constraints, two of the most common and powerful constructs in ASP. In the above systems, atoms that are always false are eliminated by the solver during the grounding phase. To address the limitation, I proposed xASP [48, 49], which introduces grounding as-needed process. xASP computes only the set of ground rules that are necessary for the construction related to the query atom, without simplifying the program before finding the explanations. Thus, this approach provides faithful explanations for the truth value of the given atom. This is important to form a correct understanding of programs.

Handling language extensions in ASP. ASP introduces several language extensions, including aggregate like *#min* and *#max*. In our follow-up work, xASP2 [50, 51], we further improved the system by incorporating meta-programming techniques, which boosted performance through faster computation of minimal assumption sets and better handling of complex linguistic structures. This version also introduces support for extended languages with aggregation functions and includes a web-based application for visualization. The system has been empirically validated through testing in a large-scale commercial application.

Table 1 summarizes compared features: whether the explanation is guaranteed to be acyclic; whether the input program may include aggregates and constraints; whether the query atom can be false in the

Table 1
Summary of compared features [50]

System (if any) and reference	Acyclic explanation	Linguistic extensions	Explanation for false atoms	System availability
xclingo [22]	Yes	None	No	Yes
s(CASP) [25]	Yes	Constraints	Yes	Yes
Visual-DLV [52]	Yes	Constraints	No	Yes
spock [28]	Yes	Constraints	No	Yes
DWASP [30]	Yes	Constraints	No	Yes
[26]	No	None	Yes	No
[27]	Yes	None	Yes	No
ASPeRiX [53]	Yes	Constraints	Yes	Yes
LABAS [54]	No	None	Yes	Yes
[55]	No	Aggregates	Yes	No
xASP2	Yes	Aggregates and Constraints	Yes	Yes

answer set; and whether the system is available for experimentation. Further details on these differences are provided in the work [50].

5.2. XAI for Deep Neural Network

To address explainability in DNN, we propose xDNN$^{(ASP)}$ [56]–a novel framework for generating global explanations of DNNs by extracting logic-based representations under ASP. This work falls within the taxonomy of post-hoc explanation methods, which analyze trained models to uncover their decision-making processes. It aims to answer the key questions: (1) "Why is the output produced?", (2) "What features were involved?", and (3) "How was the interaction among the hidden layers?"

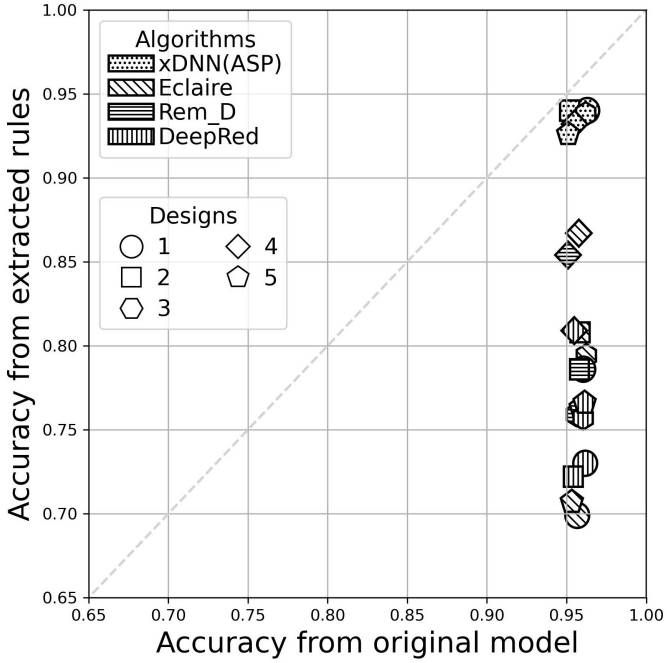

Figure 1: Comparing xDNN$^{(ASP)}$, ECLAIRE, REM-D and DeepRED [56]

Given a trained neural network and its associated training data, xDNN$^{(ASP)}$ derives a logic program whose answer sets – in the ideal case – represents the trained model, i.e., answer sets of the extracted program correspond one-to-one to input-output pairs of the network. xDNN$^{(ASP)}$ has been applied in two synthetic datasets, XOR operator dataset and its variation, using five different configurations of network design. Fig. 1 compares the accuracy of NNs and four rule extraction algorithms across five designs, while Fig. 2 presents fold-level accuracy for NN, ECLAIRE and xDNN$^{(ASP)}$ on design 2. If the extracted rules

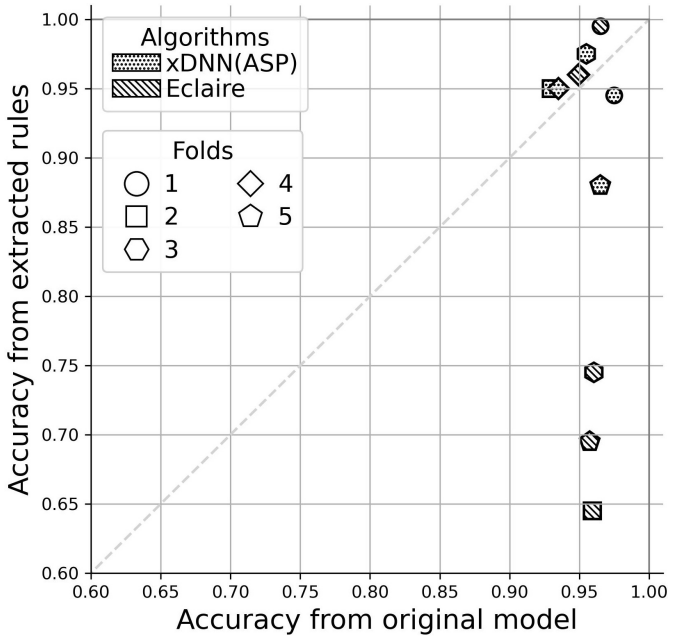

Figure 2: xDNN$^{(\text{ASP})}$ vs. ECLAIRE wrt. 5-fold in Design 2 [56]

closely mirror the behavior of their original NN models, their accuracies should align closely, meaning that each point should lie near the diagonal line. In Fig. 1, while all NN models performed well across the five designs, the rule accuracy of xDNN$^{(\text{ASP})}$ consistently outperformed the others, demonstrating high fidelity. Fig. 2 shows that, while ECLAIRE reached 99.5% accuracy in one fold, its performance dropped to about 65% in another, indicating inconsistency within the same design. Thus, ECLAIRE may need more exploration of NN designs to improve both rule extraction and model accuracy. In contrast, xDNN$^{(\text{ASP})}$ delivered more stable results across all folds, despite slightly lower peak accuracy. In addition to maintaining high predictive accuracy, the extracted logic program offers insightful information about the model, such as feature importance and the influence of hidden nodes on predictions. The latter can be used as a guide for reducing the number of nodes used in hidden layers, i.e., providing a means for optimizing the network. More experimental details and results are available in the work [56].

6. Conclusion, Open Issues and Future Work

The latest work, xASP2 , is an xAI system designed for ASP, capable of explaining the truth or falsity of atoms in an answer set. It supports advanced clingo constructs such as aggregates and constraints. Additionally, we introduce xDNN$^{(\text{ASP})}$, which extracts ASP representations from trained deep neural networks to provide global explanations. While preserving predictive accuracy, it offers insights into feature importance and the influence of hidden nodes, aiding in model interpretation and potential network optimization. Although the size of the generated logic program scales linearly with the network, it remains useful for analysis and explanation. Future research could focus on reducing computational costs. One potential approach is to examine the impact of groups of hidden nodes rather than individual ones. Another interesting direction is to explore specialized neural network architectures, such as CNNs, to tackle classification problems involving non-numerical data, such as images.

Acknowledgments

I would like to thank my advisor, Dr. Son Tran, for his encouragement, interest, support, guidance and patience. I would also like to thank the Harold James Family Trust and Dr. Derek Bailey for supporting me during my PhD studies. The research is partially supported by Los Alamos National Laboratory under the award number 047229.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, P. M. Atkinson, Explainable artificial intelligence: an analytical review, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 11 (2021) e1424.

[2] W. Samek, K.-R. Müller, Towards explainable artificial intelligence, Explainable AI: interpreting, explaining and visualizing deep learning (2019) 5–22.

[3] S. M. McKinney, M. Sieniek, V. Godbole, J. Godwin, N. Antropova, H. Ashrafian, T. Back, M. Chesus, G. S. Corrado, A. Darzi, et al., International evaluation of an ai system for breast cancer screening, Nature 577 (2020) 89–94.

[4] S. Graham, C. Depp, E. E. Lee, C. Nebeker, X. Tu, H.-C. Kim, D. V. Jeste, Artificial intelligence for mental health and mental illnesses: an overview, Current psychiatry reports 21 (2019) 1–18.

[5] L. Pantanowitz, G. M. Quiroga-Garza, L. Bien, R. Heled, D. Laifenfeld, C. Linhart, J. Sandbank, A. A. Shach, V. Shalev, M. Vecsler, et al., An artificial intelligence algorithm for prostate cancer diagnosis in whole slide images of core needle biopsies: a blinded clinical validation and deployment study, The Lancet Digital Health 2 (2020) e407–e416.

[6] A. W. Sadek, Artificial intelligence applications in transportation, Transportation research circular (2007) 1–7.

[7] D. Gunning, D. Aha, Darpa's explainable artificial intelligence (xai) program, AI magazine 40 (2019) 44–58.

[8] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.-Z. Yang, Xai—explainable artificial intelligence, Science robotics 4 (2019) eaay7120.

[9] W. Saeed, C. Omlin, Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities, Knowledge-Based Systems 263 (2023) 110273. URL: https://www.sciencedirect.com/science/article/pii/S0950705123000230. doi:https://doi.org/10.1016/j.knosys.2023.110273.

[10] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, F. Herrera, Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence, Information Fusion 99 (2023) 101805.

[11] P. Gohel, P. Singh, M. Mohanty, Explainable ai: current status and future directions, arXiv preprint arXiv:2107.07045 (2021).

[12] J. Fandinno, C. Schulz, Answering the "why" in answer set programming–a survey of explanation approaches, Theory and Practice of Logic Programming 19 (2019) 114–203.

[13] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM computing surveys (CSUR) 51 (2018) 1–42.

[14] G. Vilone, L. Longo, Notions of explainability and evaluation approaches for explainable artificial intelligence, Information Fusion 76 (2021) 89–106.

[15] B. Goodman, S. Flaxman, European union regulations on algorithmic decision-making and a "right to explanation", AI magazine 38 (2017) 50–57.

[16] V. Marek, M. Truszczyński, Stable models and an alternative logic programming paradigm, in: The Logic Programming Paradigm: a 25-year Perspective, 1999, pp. 375–398. doi:10.1007/978-3-642-60085-2_17.

[17] I. Niemelä, Logic programming with stable model semantics as a constraint programming paradigm, Annals of Mathematics and Artificial Intelligence 25 (1999) 241–273. doi:10.1023/A:1018930122475.

[18] M. Gelfond, V. Lifschitz, Logic programs with classical negation, in: D. Warren, P. Szeredi (Eds.), 7th ICLP, 1990, pp. 579–597.
[19] M. Gebser, B. Kaufmann, A. Neumann, T. Schaub, clasp: A conflict-driven answer set solver, in: C. Baral, G. Brewka, J. Schlipf (Eds.), Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07), volume 4483 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2007, pp. 260–265.
[20] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[21] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, volume 4, Springer, 2006.
[22] P. Cabalar, J. Fandinno, B. Muñiz, A system for explainable answer set programming, Electronic Proceedings in Theoretical Computer Science 325 (2020) 124–136.
[23] P. Cabalar, B. Muñiz, Explanation graphs for stable models of labelled logic programs, in: J. Arias, S. Batsakis, W. Faber, G. Gupta, F. Pacenza, E. Papadakis, L. Robaldo, K. Rückschloß, E. Salazar, Z. G. Saribatur, I. Tachmazidis, F. Weitkämper, A. Z. Wyner (Eds.), Proceedings of the International Conference on Logic Programming 2023 Workshops co-located with the 39th International Conference on Logic Programming (ICLP 2023), London, United Kingdom, July 9th and 10th, 2023, volume 3437 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3437/paper3ASPOCP.pdf.
[24] P. Cabalar, B. Muñiz, Model explanation via support graphs, Theory Pract. Log. Program. (2024). doi:10.1017/S1471068424000048.
[25] J. Arias, M. Carro, Z. Chen, G. Gupta, Justifications for goal-directed constraint answer set programming, Electronic Proceedings in Theoretical Computer Science 325 (2020) 59–72.
[26] E. Pontelli, T. C. Son, O. Elkhatib, Justifications for logic programs under answer set semantics, Theory and Practice of Logic Programming 9 (2009) 1–56.
[27] C. Viegas Damásio, A. Analyti, G. Antoniou, Justifications for logic programming, in: Logic Programming and Nonmonotonic Reasoning: 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proc. 12, Springer, 2013, pp. 530–542.
[28] M. Brain, M. Gebser, J. Pührer, T. Schaub, H. Tompits, S. Woltran, That is illogical captain! the debugging support tool spock for answer-set programs: system description, in: Proceedings of the Workshop on Software Engineering for Answer Set Programming (SEA'07), 2007, pp. 71–85.
[29] J. P. Delgrande, T. Schaub, H. Tompits, A framework for compiling preferences in logic programs, Theory and Practice of Logic Programming 3 (2003) 129–187.
[30] C. Dodaro, P. Gasteiger, K. Reale, F. Ricca, K. Schekotihin, Debugging non-ground asp programs: Technique and graphical tools, Theory and Practice of Logic Programming 19 (2019) 290–316.
[31] M. Gebser, R. Kaminski, A. König, T. Schaub, Advances in gringo series 3, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2011, pp. 345–351.
[32] M. Alviano, C. Dodaro, N. Leone, F. Ricca, Advances in wasp, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2015, pp. 40–54.
[33] P. Biecek, T. Burzykowski, Local interpretable model-agnostic explanations (LIME), EMAE, Explain and Examine Predictive Models 1 (2021) 107–124.
[34] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).
[35] R. Guidotti, Counterfactual explanations and how to find them: literature review and benchmarking, Data Mining and Knowledge Discovery 38 (2024) 2770–2824.
[36] J. Ferreira, M. de Sousa Ribeiro, R. Goncalves, J. Leite, Looking inside the black-box: Logic-based explanations for neural networks, in: Proceedings of the international conference on principles of knowledge representation and reasoning, volume 19, 2022, pp. 432–442.
[37] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-based systems 8 (1995) 373–389.
[38] J. R. Zilke, E. Loza Mencía, F. Janssen, Deepred–rule extraction from deep neural networks, in: Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings 19, Springer, 2016, pp. 457–473.

[39] M. E. Zarlenga, Z. Shams, M. Jamnik, Efficient decompositional rule extraction for deep neural networks, arXiv preprint arXiv:2111.12628 (2021).
[40] E. W. Saad, D. C. Wunsch II, Neural network explanation using inversion, Neural networks 20 (2007) 78–93.
[41] M. G. Augasta, T. Kathirvalavakumar, Reverse engineering the neural networks for rule extraction in classification problems, Neural processing letters 35 (2012) 131–150.
[42] E. R. Hruschka, N. F. Ebecken, Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach, Neurocomputing 70 (2006) 384–397.
[43] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to data mining, Pearson., 2016.
[44] F. Pedregosa, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
[45] L. L. Trieu, T. C. Son, E. Pontelli, M. Balduccini, Generating explanations for answer set programming applications, in: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III, volume 11746, SPIE, 2021, pp. 390–403.
[46] B. N. Hanna, L. L. T Trieu, T. C. Son, N. T. Dinh, An application of asp in nuclear engineering: Explaining the three mile island nuclear accident scenario, Theory and Practice of Logic Programming 20 (2020) 926–941.
[47] L. L. Trieu, T. C. Son, M. Balduccini, exp(aspc): Explaining asp programs with choice atoms and constraint rules, EPTCS 345 (2021) 155–161. URL: http://dx.doi.org/10.4204/EPTCS.345.28. doi:10.4204/eptcs.345.28.
[48] L. L. Trieu, T. C. Son, M. Balduccini, xasp: An explanation generation system for answer set programming, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2022, pp. 363–369.
[49] M. Alviano, L. L. T. Trieu, T. C. Son, M. Balduccini, Advancements in xasp, an XAI system for answer set programming, in: A. Dovier, A. Formisano (Eds.), Proceedings of the 38th Italian Conference on Computational Logic, Udine, Italy, June 21-23, 2023, volume 3428 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3428/paper2.pdf.
[50] M. Alviano, L. Ly Trieu, T. Cao Son, M. Balduccini, The xai system for answer set programming xasp2, Journal of Logic and Computation 34 (2024) 1500–1525.
[51] M. Alviano, L. L. Trieu, T. Cao Son, M. Balduccini, Explanations for answer set programming, EPTCS 385 (2023) 27–40. URL: http://dx.doi.org/10.4204/EPTCS.385.4. doi:10.4204/eptcs.385.4.
[52] S. Perri, F. Ricca, G. Terracina, D. Cianni, P. Veltri, An integrated graphic tool for developing and testing dlv programs, in: Proceedings of the Workshop on Software Engineering for Answer Set Programming (SEA'07), 2007, pp. 86–100.
[53] C. Béatrix, C. Lefèvre, L. Garcia, I. Stéphan, Justifications and blocking sets in a rule-based answer set computation, in: Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016, pp. 6:1–6:15.
[54] C. Schulz, F. Toni, Justifying answer sets using argumentation, Theory and Practice of Logic Programming 16 (2016) 59–110.
[55] S. Marynissen, J. Heyninck, B. Bogaerts, M. Denecker, On nested justification systems (full version), arXiv preprint arXiv:2205.04541 (2022).
[56] L. L. Trieu, T. Cao Son, xdnn(asp): Explanation generation system for deep neural networks powered by answer set programming, *Accepted* in The 41st International Conference on Logic Programming, 2025.