Advancements in xASP

Mario Alviano[1], Ly Ly Trieu[2], Tran Son[2], and Marcello Balduccini[3]

[1] DEMACS, University of Calabria, Via Bucci 30/B, 87036 Rende (CS), Italy
mario.alviano@unical.it
[2] New Mexico State University, USA
lytrieu@nmsu.edu, tson@cs.nmsu.edu
[3] Saint Joseph's University, USA
mbalducc@sju.edu

Extended Abstract

The interest in explainable artificial intelligence (XAI) has grown substantially in recent years in the light of the fact that confidence in solutions provided by intelligent systems requires the possibility to query them about the reasoning process. Answer Set Programming (ASP) [9, 11] is not an exception, and explainability in this setting can be formulated as the following question: *Given an answer set A of a program Π and an atom α, why does $\alpha \in A$ (or $\alpha \notin A$)?* As a logic program Π is a set of rules, the question can be answered by providing the subset of Π that supports the presence (or the absence) of α given Π and A. If Π is a Datalog program, then its models are easily explainable by the derivation procedure implemented by Datalog engines. Essentially, each atom in the model is explained by the support provided by a rule whose body is true and contains only already explained atoms. If Π is a logic program under the well-founded semantics, then the fact that α belongs (or does not belong) to the well-founded model of Π can be explained similarly, with the addition of some atoms that are concluded to be false because they belong to some unfounded set. Generally speaking, explanations for logic programs under the answer set semantics can also be produced in a similar way *under the assumption provided by the answer sets themselves for the interpretation of false atoms*. However, taking all false atoms as an assumption would likely result in a *faint* explanation, actually in an explanation by faith for all such false atoms.

In order to extend the explainability of Datalog to broader fragments of ASP, three main issues need to be tackled in explaining the assignment of α in A:

(i) How to compute a hopefully small set of assumptions capable of explaining the assignment of α in A. Such a set takes the name of *minimal assumption set*, and can be defined as a minimal set of false atoms sufficient to reconstruct the answer set by applying some inference rules that mimic those implemented by ASP engines but are simpler to explain. The reconstructing sequence itself constitutes an *explaining derivation* from which a directed acyclic graph (DAG) can be synthesized.

(ii) How to handle constraints and rules acting as constraints, which can be taken into account to explain the falsity of some atoms in easily understandable terms. Indeed, while truth of any atom β in an answer set *must be supported* by at least one rule whose body is true and whose head contains β, such a support could emerge only after some other atoms are concluded false due to some constraints or, more in general, due to a rule whose head is false.

(iii) How to deal with other expressive constructs of ASP such as aggregates and choice rules. Aiming at obtaining simple explanations, and restricting to the most common case of stratified aggregates, the truth value of an aggregate can be considered derived once all atoms in its aggregate set are explained. For the purpose of explainability, aggregates can therefore be rewritten

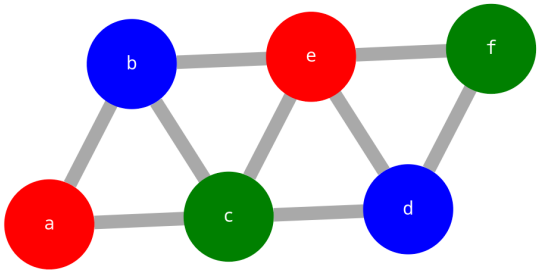

Figure 1: The undirected graph and its 3-coloring used in Example 1

in terms of simple rules. Regarding choice rules, in addition to the support they may naturally provide to true atoms, they can impose the falsity of some atoms due to their upper bound; in fact, once the number of true head atoms in a choice rule (whose body is true) reaches its upper bound, the other head atoms must be false.

We addressed the above issues in our recent work [1, 2] leading to the second version of the XAI system `xASP` [16]. The system is powered by the `clingo python api` [8]. It takes as input an ASP program Π, one of its answer sets A, and an atom α, and can produce in output minimal assumption sets, explaining derivations, and DAGs to help the user figure out the assignment of α. The source code is available at https://github.com/alviano/xasp and an interactive navigator for DAGs is hosted at https://xasp-navigator.netlify.app/. In a nutshell, the pipeline implemented by `xASP2` starts by serializing the input data, that is, by representing the input program Π, the answer set A, and the query atom α in terms of facts. The serialization itself is obtained by means of an ASP program crafted from the abstract syntax tree of Π and whose unique answer set identifies the relevant portion of the ground expansion of Π. After that, `xASP2` proceeds by computing a minimal assumption set, an explaining derivation and an explanation DAG by means of ASP programs. As an additional optimization, the explaining derivation is shrunk to the atoms reachable from α, again by means of an ASP program. Finally, the user can opt for a few additional steps: obtain a graphical representation by means of the `igraph` network analysis package (https://igraph.org/); obtain an interactive representation in https://xasp-navigator.netlify.app/; ask for different minimal assumption sets, explaining derivations and DAGs.

Example 1 (Graph 3-Colorability)**.** Let Π_{run} comprise rules (1)–(4) below, assigns a color among *red*, *green* and *blue* to each node so that adjacent nodes have different colors, and facts over *node*/1 and *edge*/2 encoding the undirected graph in Figure 1.

$$assign(X, red) \leftarrow node(X),\ not\ assign(X, green),\ not\ assign(X, blue) \tag{1}$$
$$assign(X, green) \leftarrow node(X),\ not\ assign(X, red),\ not\ assign(X, blue) \tag{2}$$
$$assign(X, blue) \leftarrow node(X),\ not\ assign(X, green),\ not\ assign(X, red) \tag{3}$$
$$\bot \leftarrow edge(X, Y),\ assign(X, C),\ assign(Y, C) \tag{4}$$

Among the answer sets of program Π_{run} there is A_{run} shown in Figure 1, containing, among others, the atoms $assign(a, red)$, $assign(b, blue)$, and $assign(c, green)$. The system `xASP2` can explain the falsity of $assign(e, green)$ by providing the DAG shown in Figure 2. ■

An XAI system providing the reasons for the presence or absence of a given atom in an answer set finds another important application in the identification of the cause of unexpected results. This is a feature that can be particularly useful to the designers of complex systems confronted with unexpected inferences. In fact, identifying the root causes of those inferences can be daunting due to the

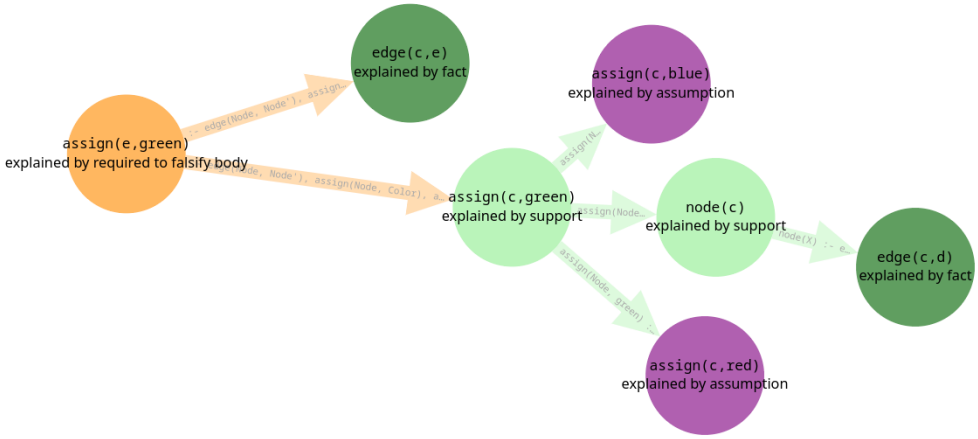

Figure 2: Induced DAG on the vertices reachable from $assign(e, green)$ for the minimal assumption set $\{assign(c, red), assign(c, blue), assign(e, red)\}$ for Π_{run}. Note that the assumption $assign(e, red)$ is not used in the portion of the DAG explaining $assign(e, green)$.

many possible interactions in large knowledge bases. Table 1 summarizes a comparison with both debugging tools for ASP and cutting-edge XAI systems designed for ASP. The compared features are the following: whether the explanation is guaranteed to be acyclic; the capability to handle the input program with aggregates and constraints; the ability to provide an explanation when the query atom can be false in the answer set; and whether the system is available for experimentation. As can be seen from Table 1, our system is capable of providing explanations for false atoms and does not lead to cyclic argumentation in the explanation. `xASP2` is the only system that tackles a program that includes both aggregates and constraints. It is worth noting that the topic of aggregates is addressed in another approach [10], even though no system implementing this approach is mentioned or available.

In conclusion, we have developed and implemented `xASP2` , an XAI system that targets the ASP language and is powered by ASP engines. Our approach to explaining why an atom is true or false in an answer set involves deriving easy-to-understand inferences originating from a hopefully small

Table 1: Summary of compared features

System (if any) and reference	Acyclic explanation	Linguistic extentions	Explanation for false atoms	System availability
`s(CASP)` [3]	Yes	Constraints	Yes	Yes
`ASPeRiX` [4]	Yes	Constraints	Yes	Yes
`spock` [5]	Yes	Constraints	No	Yes
`xclingo` [6]	Yes	None	No	Yes
`DWASP` [7]	Yes	Constraints	No	Yes
[10]	No	Aggregates	Yes	No
`Visual-DLV` [12]	Yes	Constraints	No	Yes
[13]	No	None	Yes	No
`LABAS` [14]	No	None	Yes	Yes
`exp(ASP`^{c}`)` [15]	No	Constraints	Yes	Yes
[17]	Yes	None	Yes	No
`xASP2`	Yes	Aggregates and Constraints	Yes	Yes

set of atoms assumed false. `xASP2` has the ability to support different `xclingo` constructs such as aggregates and constraints. An explanations is produced in the form of a DAG with the atom to be explained as the root.

Acknowledgments

Portions of this publication and research effort are made possible through the help and support of NIST via cooperative agreement 70NANB21H167. Tran Cao Son was also partially supported by NSF awards #1757207 and #1914635. Mario Alviano was also partially supported by Italian Ministry of Research (MUR) under PRIN project PRODE “, under PNRR project FAIR “Future AI Research”, CUP H23C22000860006, under PNRR project Tech4You “Technologies for climate change adaptation and quality of life improvement”, CUP H23C22000370006, and under PNRR project SERICS “SEcurity and RIghts in the CyberSpace”, CUP H73C22000880001; by Italian Ministry of Health (MSAL) under POS project RADIOAMICA, CUP H53C22000650006; by the LAIA lab (part of the SILA labs) and by GNCS-INdAM.

References

[1] Mario Alviano, Ly Ly T. Trieu, Tran Cao Son, and Marcello Balduccini. Advancements in xasp, an XAI system for answer set programming. In Agostino Dovier and Andrea Formisano, editors, *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.

[2] Mario Alviano, Ly Ly T. Trieu, Tran Cao Son, and Marcello Balduccini. Explanations for answer set programming. *CoRR*, abs/2308.15879, 2023.

[3] Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. Justifications for goal-directed constraint answer set programming. *Electronic Proceedings in Theoretical Computer Science*, 325:59–72, Sep 2020.

[4] Christopher Béatrix, Claire Lefèvre, Laurent Garcia, and Igor 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.

[5] Martin Brain, Martin Gebser, Jörg Pührer, Torsten Schaub, Hans Tompits, and Stefan 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)*, pages 71–85, 2007.

[6] Pedro Cabalar, Jorge Fandinno, and Brais Muñiz. A system for explainable answer set programming. *Electronic Proceedings in Theoretical Computer Science*, 325:124–136, Sep 2020.

[7] Carmine Dodaro, Philip Gasteiger, Kristian Reale, Francesco Ricca, and Konstantin Schekotihin. Debugging non-ground asp programs: Technique and graphical tools. *Theory and Practice of Logic Programming*, 19(2):290–316, 2019.

[8] Roland Kaminski, Javier Romero, Torsten Schaub, and Philipp Wanko. How to build your own asp-based system?! *Theory and Practice of Logic Programming*, 23(1):299–361, 2023.

[9] V. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-year Perspective*, pages 375–398, 1999.

[10] Simon Marynissen, Jesse Heyninck, Bart Bogaerts, and Marc Denecker. On nested justification systems (full version). *arXiv preprint arXiv:2205.04541*, 2022.

[11] I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.

[12] Simona Perri, Francesco Ricca, Giorgio Terracina, D Cianni, and 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)*, pages 86–100, 2007.

[13] Enrico Pontelli, Tran Cao Son, and Omar Elkhatib. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming*, 9(1):1–56, 2009.

[14] Claudia Schulz and Francesca Toni. Justifying answer sets using argumentation. *Theory and Practice of Logic Programming*, 16(1):59–110, 2016.

[15] Ly Ly Trieu, Tran Cao Son, and Marcello Balduccini. exp(aspc): explaining asp programs with choice atoms and constraint rules. *Electronic Proceedings in Theoretical Computer Science*, 2021.

[16] Ly Ly Trieu, Tran Cao Son, and Marcello Balduccini. xasp: An explanation generation system for answer set programming. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 363–369. Springer, 2022.

[17] Carlos Viegas Damásio, Anastasia Analyti, and Grigoris Antoniou. Justifications for logic programming. In *Logic Programming and Nonmonotonic Reasoning: 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proc. 12*, pages 530–542. Springer, 2013.