Enhancing the Extensibility of Answer Set Programming via Meta-Programming

Amadé Nemes[1]

[1]University of Potsdam, Germany, Institute of Computer Science, An der Bahn 2, 14476 Potsdam

1. Introduction

Answer Set Programming (ASP; [1]) has solidified itself as a leading logic programming paradigm for solving complex, knowledge intensive tasks. This is due to the attractive combination of efficient solver implementations such as the *dlv* [2] and *clingo* [3] systems, as well as the clear logical foundation for program semantics provided by the Logic of Here-and-There and its nonmonotonic extension, Equilibrium Logic (EL; [4]).

The discovery of this logical specification of the semantics of ASP has led to a burgeoning of new language extensions, enhancing the language with hybrid constraints [5] and temporal [6], deontic [7], metric [8][9] and dynamic [10] [11] modalities, to name a few. This stems from the relative ease with which one can construct a non-monotonic variant of classical logical systems, such as Linear-time Temporal Logic (LTL [12]), Linear-time Dynamic Logic (LDL [13]), and Metric Temporal Logic (MTL [14]), by adding the appropriate minimization criterion to obtain an equilibrium nonmonotonic variant of the given logic, yielding in these cases Temporal Equilibrium Logic (TEL), Dynamic Equilibrium Logic (DEL), and Metric Equilibrium Logic (MEL), respectively.

On the other hand, developing and maintaining systems that implement these logics can prove challenging. Answer set solvers are highly optimized system and thus are designed for efficiency first, and ease of extensibility second, hindering the exploration of new ASP extensions. Nonetheless, the *clingo* system does provide a rich API and support for language extensions via it's theory grammar specifications [15], which provides all tools necessary for skilled programmers familiar with the interface. Of the aforementioned extensions to ASP, TEL + DEL, Deontic Equilibrium Logic and ASP with hybrid constraints have been realized in form of *telingo* [16], *deolingo* [7], and *flingo*[17], respectively, all of which rely on the theory capabilities of clingo.

Still, these systems are a significant undertaking to implement, as evidenced by the delay between theory and implementation and the complete lack of a system in some cases. Furthermore these systems are difficult to modify or combine. Indeed, the combination of hybrid constraints and temporal modalities has already seen preliminary investigation [18]; the combination of the *flingo* and *telingo* systems however is a difficult and complicated undertaking.

The central hypothesis of my research topic is that the implementation and extensibility gap can be addressed by applying meta-programming techniques.

2. Background

The application of meta-programming for extensibility has a rich history in the area of Artificial Intelligence and Knowledge Representation and Reasoning, stretching back to McCarthy and the LISP programming language [19]. The logic programming language Prolog [20] in particular, due to it's interpreted nature, can boast a dizzying amount of meta-interpreters that extend the core language.

Joint Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026) co-located with the Federated Logic Conference (FLoC 2026).

amade.nemes@uni-potsdam.de (A. Nemes)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-03

CEUR-WS.org/Vol-4311/DC_short_10.pdf

In contrast, meta-programming techniques in ASP [21, 22, 23, 24, 25, 26, 27, 28], while not unexplored, is relatively speaking seldom applied. One might argue that this is due to the difference in architecture of ASP systems and Prolog. The interpreted and homoiconic nature of Prolog allows user to implement a meta-interpreter for Prolog in a few lines of code, and ensures these meta-interpreters remain malleable. In contrast, established ASP systems apply a ground-and-solve approach. Thus, for example, the meta-programming support provided by *clingo* relies on first grounding a program, then reifying the ground logic program's syntax as a set of facts. These facts can then be combined with a meta-encoding that re-interprets the ground language constructs, assigning them new semantics by passing both to clingo to re-ground and solve.

While this approach is also amenable to create succinct meta-interpreters for the core language, as evidenced in [27], things become more and more complicated the further one strays from the syntax and semantics of plain ASP. First, established implementation techniques in systems such as *telingo* require pre-processing of an input program before instantiation, a facility which current meta-programming approaches do not provide. Second, ASP grounders such as gringo are heavily optimized, performing many semantics-preserving program simplifications [29] during grounding. Grounders such as gringo also have no mechanism to be informed of the semantics any given meta-interpreter is implementing, and as such may (and often do) perform simplifications that are not semantics-preserving, jeopardizing the correctness of the implementation. While such issues can in some cases be mitigated by clever use of external statements [15] in the case of gringo, the correct way applying such strategies can be intricate, and in some cases unclear.

These issues surrounding semantics do not just come from technical limitations, but rather also from a lack of theoretical exploration of the first-order semantics for extensions of ASP. While such a semantics has been established for some time for regular ASP in the form of Quantified Equilibrium Logic [30] [31], the same work on the first-order level has not been seen through, or have only been preliminarily explored [32] for the various extensions of EL we have listed. Due to this gap in theoretical understanding, the correct end-to-end behaviour of systems implementing the full logical systems, in particular how correct and efficient grounding should be performed, remains unclear in most cases. Furthermore, the theoretical underpinnings and of meta-interpreters have not been studied, and as such the correctness of even the simplest meta-interpreters implementing regular ASP was not established.

My research topic focuses on tackling both technical and theoretical challenges surrounding meta-programming for ASP, by providing new systems to facilitate meta-programming, expanding its usage to new application domains, studying first order semantics for extensions of ASP and theoretical properties of meta-interpreters.

3. Research

The preliminary parts of my research into meta-programming for ASP are encapsulated in my Master's thesis [33] on the topic. Here I presented a meta-interpreter for the full unrestricted language of TEL allowing for nesting of arbitrary temporal modalities, thus forgoing the restriction to past-future rules that is present in *telingo*. I furthermore characterized the meta-interpreter as implementing the semantics as a Tseitin-style transformation, and proved that this transformation provides a one-to-one correspondence between the stable models of the meta-interpreter, and the temporal stable models of the underlying theory.

Going further, I expanded the discussion to the non-ground level, defining a fully unrestricted variant of Quantified Temporal Equilibrium Logic (QTEL), extending the version defined in [32] to handle arbitrary temporal theories. For this extended variant of QTEL I provided an extended definition of safety for nested temporal modalities based on reachability through safe arguments of modal operators, and proved that the usual desirable properties of Domain-Independence and No Unnamed Individuals [34] [35] hold for this extended definition of safety. Finally, by exploiting this new safety definition, I provided a method of grounding temporal programs using external statements which, while still guaranteeing semantic correctness, could significantly reduce the ground instantiation of the program

when compared to naive grounding. In future work, I plan on expanding my investigation into the theoretical properties of First-Order variants of other extensions of Equilibrium Logic.

In more recent work, I have tackled the lack of a general-purpose meta-programming framework for non-ground ASP programs by developing *aspen-tree* [36] [37], a novel system specifically designed for the reification, transformation, and analysis of non-ground ASP programs. Built upon the widely used *tree-sitter* [1] parsing library, *aspen-tree* provides a language-independent and iterative framework for mapping source code to a reified fact base. These facts are then processed by *clingo* using meta-encodings to provide static analysis or to generate high-level transformation instructions, which are applied via a templating engine to update the source code.

Notably, as the system is not built with a language specific architecture, but rather functions for any programming language with an associated *tree-sitter* grammar, it can provide its functionality for any such language, with the number of such languages being in the hundreds. This opens up exciting future research opportunities for applying ASP for static analysis and optimization of programming languages, an area where ASP has not seen much application, despite the success of the related language Datalog.

Demonstrating its practical viability, *aspen-tree* already serves as a core component of the *metasp* framework [38, 28], a holistic system designed to extend and enhance the built-in meta-programming support in *clingo*, where *aspen-tree* drives a preprocessing pipeline to extend *clingo* with complex new language constructs. *metasp* extends standard ASP meta-programming by augmenting *clingo*'s theory grammar with formal type specifications for nested modalities. To ensure semantic correctness, the system also allows the specification of safety properties of modalities, and based on this specification constructs a transformation pipeline that protects nested modalities from stable-model-based simplifications during grounding. We provide complete meta-encodings implementing TEL, DEL and MEL using the facilities provided by *metasp*.

4. Declaration on Generative AI

During the preparation of this work, the author(s) used Claude Sonnet 5 to identify and correct grammatical errors, typos, and other writing mistakes. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] V. Lifschitz, Answer Set Programming, Springer-Verlag, 2019. doi:10.1007/978-3-030-24658-7.

[2] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello, The DLV system for knowledge representation and reasoning, ACM Transactions on Computational Logic 7 (2006) 499–562.

[3] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82. doi:10.1017/S1471068418000054.

[4] D. Pearce, Equilibrium logic, Annals of Mathematics and Artificial Intelligence 47 (2006) 3–41. doi:10.1007/s10472-006-9028-z.

[5] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, Towards a semantics for hybrid ASP systems, CoRR abs/2108.03061 (2021). URL: https://arxiv.org/abs/2108.03061.

[6] P. Cabalar, R. Kaminski, T. Schaub, A. Schuhmann, Temporal answer set programming on finite traces, Theory and Practice of Logic Programming 18 (2018) 406–420. doi:10.1017/S1471068418000297.

[7] P. Cabalar, O. Manteiga, deolingo: Extending answer set programming with deontic reasoning (2025) 61–77.

[8] P. Cabalar, M. Diéguez, T. Schaub, A. Schuhmann, Towards metric temporal answer set programming, Theory and Practice of Logic Programming 20 (2020) 783–798.

[1] https://tree-sitter.github.io/tree-sitter

[9] P. Cabalar, M. Diéguez, T. Schaub, A. Schuhmann, Metric temporal answer set programming over timed traces, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), Proceedings of the Sixteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'22), volume 13416 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2022, pp. 117–130. doi:10.1007/978-3-031-15707-3_10.

[10] P. Cabalar, M. Diéguez, T. Schaub, Towards dynamic answer set programming over finite traces, in: [39], 2019, pp. 148–162. doi:10.1007/978-3-030-20528-7_12.

[11] P. Cabalar, M. Diéguez, F. Laferriere, T. Schaub, Implementing dynamic answer set programming over finite traces, in: G. De Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20), IOS Press, 2020, pp. 656–663. doi:10.3233/FAIA200151.

[12] A. Pnueli, The temporal logic of programs, in: Proceedings of the Eighteenth Symposium on Foundations of Computer Science (FOCS'77), IEEE Computer Society Press, 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.

[13] G. De Giacomo, M. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: F. Rossi (Ed.), Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI'13), IJCAI/AAAI Press, 2013, pp. 854–860.

[14] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Systems 2 (1990) 255–299. doi:10.1007/BF01995674.

[15] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: M. Carro, A. King, N. Saeedloei, M. De Vos (Eds.), Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16), volume 52 of *Open Access Series in Informatics (OASIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:15.

[16] P. Cabalar, R. Kaminski, P. Morkisch, T. Schaub, telingo = ASP + time, in: [39], 2019, pp. 256–269. doi:10.1007/978-3-030-20528-7_19.

[17] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, Flingo – instilling asp expressiveness into linear integer constraints, 2026.

[18] P. Cabalar, M. Diéguez, F. Olivier, T. Schaub, I. Stéphan, Temporal here-and-there with constraints (2023).

[19] J. McCarthy, History of lisp, in: History of Programming Languages, 1978, pp. 173–185.

[20] A. Colmerauer, An introduction to prolog iii, Communications of the ACM 33 (1990) 69–90.

[21] M. Gelfond, T. Son, Reasoning with prioritized defaults., in: J. Dix, L. Pereira, T. Przymusinski (Eds.), Third International Workshop on Logic Programming and Knowledge Representation, volume 1471 of *Lecture Notes in Computer Science*, Springer-Verlag, 1997, pp. 164–223.

[22] C. Baral, M. Gelfond, Reasoning agents in dynamic domains, in: J. Minker (Ed.), Logic-Based Artificial Intelligence, Kluwer Academic Publishers, Dordrecht, 2000, pp. 257–279.

[23] T. Son, C. Baral, T. Nam, S. McIlraith, Domain-dependent knowledge in answer set planning, ACM Transactions on Computational Logic 7 (2006) 613–657. doi:10.1145/1183278.1183279.

[24] T. Eiter, A. Polleres, Towards automated integration of guess and check programs in answer set programming: a meta-interpreter and applications, Theory and Practice of Logic Programming 6 (2006) 23–60.

[25] M. Gebser, J. Pührer, T. Schaub, H. Tompits, A meta-programming technique for debugging answer-set programs, in: D. Fox, C. Gomes (Eds.), Proceedings of the Twenty-third National Conference on Artificial Intelligence (AAAI'08), AAAI Press, 2008, pp. 448–453.

[26] M. Gebser, R. Kaminski, T. Schaub, Complex optimization in answer set programming, Theory and Practice of Logic Programming 11 (2011) 821–839.

[27] R. Kaminski, J. Romero, T. Schaub, P. Wanko, How to build your own ASP-based system?!, Theory and Practice of Logic Programming 23 (2023) 299–361. doi:10.1017/S1471068421000508.

[28] S. Hahn, A. Nemes, J. Romero, T. Schaub, Meta-programming for linear-time temporal answer set programming, Accepted for publication in TPLP (2026).

[29] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Answer Set Solving in Practice, Synthesis

Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2012. doi:10.1007/978-3-031-01561-8.

[30] D. Pearce, A. Valverde, Quantified Equilibrium Logic and the First Order Logic of Here-and-There, Technical Report MA-06-02, University of Málaga, 2006.

[31] D. Pearce, A. Valverde, Quantified equilibrium logic and foundations for answer set programs, in: M. Garcia de la Banda, E. Pontelli (Eds.), Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08), volume 5366 of *Lecture Notes in Computer Science*, Springer-Verlag, 2008, pp. 546–560. doi:10.1007/978-3-540-89982-2_46.

[32] F. Aguado, P. Cabalar, G. Pérez, C. Vidal, M. Diéguez, Temporal logic programs with variables, Theory and Practice of Logic Programming 17 (2017) 226–243.

[33] A. Nemes, Implementing a Temporal Answer Set Solver using Metaprogramming Techniques, Master's thesis, University of Potsdam, 2024.

[34] P. Cabalar, D. Pearce, A. Valverde, Safety preserving transformations for general answer set programs, in: D. De Schreye (Ed.), Proceedings of the Nineteenth International Symposium on Logic Based Program Synthesis and Transformation (LOPSTR'09), volume 6037 of *Lecture Notes in Computer Science*, Springer-Verlag, 2010, pp. 132–146.

[35] P. Cabalar, D. Pearce, A. Valverde, A revised concept of safety for general answer set programs., in: E. Erdem, F. Lin, T. Schaub (Eds.), Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09), volume 5753 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2009, pp. 58–70.

[36] A. Nemes, T. Schaub, *aspen-tree*: A system for non-ground meta-programming in asp (preliminary report) (2026).

[37] A. Nemes, T. Schaub, *aspen-tree*, https://github.com/potassco/aspen-tree, 2026.

[38] S. Hahn, A. Nemes, J. Romero, T. Schaub, *metasp*, https://github.com/potassco/metasp, 2026.

[39] M. Balduccini, Y. Lierler, S. Woltran (Eds.), Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19), volume 11481 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2019.