Hybrid Answer Set Programming: Foundations and Applications

Nicolas Rühling[1]

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

Abstract

My research deals with the foundations and applications of hybrid ASP. While standard solvers like clingo have become very performant, challenges remain in regard to problems involving large numeric domains with non-trivial calculations. A wide range of hybrid ASP solvers have been introduced and successfully applied over the last couple of years. However, there are still many open questions regarding their theoretical foundations. This is needed not only to improve future implementations but also to gain a deeper understanding of the essence of the problems they are applied to. This will arguably help to find more natural ASP representations. A central application to my research is the field of (product) configuration where we have already found first results.

1. Introduction

Answer Set Programming (ASP; [1]) is being increasingly applied to solve real-world problems [2, 3, 4, 5]. However, in many cases these problems have a heterogenous nature which requires features beyond the current capacities of solvers like *clingo* [6]. Consider for example the field of configuration [7] — one of the early successful applications of ASP [8, 9]. A configuration problem usually consists of a partonomy where parts are parameterized by attributes. Both parts and their attributes can be restricted by constraints. While in simple cases these attributes are discrete, many industrial applications require attributes that range over large numeric domains (e.g., precisions in the milimeter range might be needed). Furthermore, calculations over these attributes can be of linear nature (e.g., calculating the total weight by summing up the weight of all parts) as well as non-linear (e.g., area or volume of an object, inclination of a conveyor belt, etc). Standards ASP solvers like *clingo* quickly reach their limits when dealing with numeric ranges and calculations as they need to explicitly ground all possible values. Apart from this, representing constraints in ASP that go beyond simple arithmetic expressions or aggregations generally requires considerable effort.

Over the last years, hybrid solvers such as *clingcon* [10][1] and *clingo*[DL] [11][2] have already been successfully applied to many problems such as train scheduling [12], warehouse delivery [13], and job-shop scheduling [14]. Their success lies in the use of dedicated inference methods, which allows to solve efficiently certain kinds of constraints over finite integer domains. However, what is still missing, is a solid, semantic underpinning of these systems.

This issue has first been addressed by introducing the Logic of *Here-and-There with constraints* (HT_C; [15]) as an extension of the Logic of *Here-and-There* (HT; [16]) and its non-monotone extension *Equilibrium Logic* [17]. Nowadays, HT serves as a logical foundation for ASP and has facilitated a broader understanding of this paradigm. The idea is that HT_c (and other extensions) play an analogous role for hybrid ASP. An overview of these logics is given in Section 2.

However, there remain many open questions about these logics regarding their fundamental characteristics as well as their practical use in solvers, i.e., how they can guide the implementation. Having a formal understanding of these hybrid logics is also needed to better understand the inherent structure of the (real-world) problems they are applied to and to improve their representations in ASP.

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).

nruehling@uni-potsdam.de (N. Rühling)

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

[1] https://potassco.org/clingcon

[2] https://github.com/potassco/clingo-dl

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

published 2026-11-03

CEUR-WS.org/Vol-4311/DC_short_04.pdf

2. Background

2.1. Hybrid Solvers

Nowadays, ASP solver *clingo* supports so-called *theory atoms* which allow for foreign inference methods [11, 18] following the approach of SAT *modulo theories* (SMT; [19]). The general idea is that some external theory serves as an oracle by certifying the program's stable models and has been characterized for *clingo* in [20]. Subsequently, the introduction of theory atoms has greatly facilitated the development of ASP-based special-purpose systems which make use of dedicated inference methods. Examples for this are *clingo*[DL] for the class difference constraints and *clingo*[LP] for linear programming. We proceed by giving a quick introduction of some of the hybrid solvers that are part of the POTASSCO suite.[3]

The system *clingcon* is a solver for *Constraint Answer Set Programming* (CASP) and extends the input language of *clingo* with linear equations, represented as theory atoms of the form

$$\&\texttt{sum}\{k_1 * x_1; \ldots; k_n * x_n\} \prec k_0 \tag{1}$$

where x_i is an integer variable and $k_i \in \mathbb{Z}$ an integer constant for $0 \leq i \leq n$; and $\prec$ is a comparison symbol such as $<=, =, !=, <, >, >=$. In *clingo*, theory predicates are preceded by '&'.

System *clingo*[DL] has a more restricted syntax which allows for *difference constraints* over integers. This is a subset of the syntax in (1) where theory atoms have the fixed form $\&\texttt{sum}\{\texttt{1} * x; (-\texttt{1}) * y\} <= k$ but are rewritten instead as:

$$\&\texttt{diff}\{x - y\} <= k \tag{2}$$

with x and y integer variables and $k \in \mathbb{Z}$.

A third system is *clingo*[LP][4] which extends *clingo* to solve linear constraints as dealt with in Linear Programming (LP). The syntax is identical to (1) but the domain now ranges over the real numbers. Notably, [21] contains a formal characterization of all three just mentioned systems.

Lastly, a recent addition is system *flingo* [22][5] which makes use of *clingcon* to solve ASP modulo conditional linear constraints with founded variables. While in *clingcon* all integer variables need to have a value assigned, *flingo* adds a notion of undefinedness and foundedness as known from ASP. That is, there needs to be a justification in the logic program if a variable receives a value in an answer set. Furthermore, the conditional aspect of the linear constraints can be seen as a generalization of the concept of aggregates commonly used in ASP. The syntax of *flingo* accomodates so-called *assignments* which guarantee that a variable only gets assigned a value if all other variables in its definition are itself defined, i.e., justified at some other part in the logic program. For instance, the expression

$$\&\texttt{in}\{y..y\} =: x$$

only assigns the value of y to x if y has been defined by some other rule. Omitting the assignment would permit y and x to take arbitrary values if not defined elsewhere, thereby circumventing the principle of foundedness.

Further systems not developed by POTASSCO include ASP solver *dlvhex* [23] which supports a similar concept of theory atoms as *clingo*. Other CASP systems include *dingo* [24], *mingo* [25], *ezsmt* [26], and ASP-FZN [27]. Different from the aforementioned systems, the latter four rely on translations to non-ASP solvers.

2.2. The Logic of Here-and-There and Hybrid Extensions

The logics HT and *Equilibrium Logic* nowawadays serve as a logical foundation for (plain) ASP, having brought upon fundamental results such as the notion of *strong equivalence* [28]. The idea of HT is that

[3] https://potassco.org/
[4] https://github.com/potassco/clingoLP
[5] https://github.com/potassco/flingo

of two worlds h and t, generally called *here* and *there*.[6] Both worlds are associated with sets of atoms H and T, and an *HT-interpretation* is a pair $\langle H, T\rangle$ such that $H \subseteq T$. This gives rise to a three-valued logic where atoms can either be *true*, *false* or *assumed to be true*.

The satisfaction relation for HT-interpretations is defined recursively. We only provide the base case. An atom a is *satisfied* by $\langle H, T\rangle$, in symbols $\langle H, T\rangle \models a$, iff $a \in H$, that is whenever a is satisfied at the h-world. Since we require $H \subseteq T$, this implies $a \in T$. Similarly, we can deduce that any atom that is false at the t-world, remains false at the h-world. In fact, both of these properties hold for general formulas φ in HT. The first one is called *persistence* and states that $\langle H, T\rangle \models \varphi$ implies $\langle T, T\rangle \models \varphi$. The second one states that for any negated formula we get that $\langle H, T\rangle \models \neg\varphi$ iff $\langle T, T\rangle \not\models \varphi$.

A model $\langle H, T\rangle$ of a theory Γ is called an *equilibrium model* iff (i) it is total, i.e., $H = T$, and (ii) for any H' such that $H' \subset T$, we get $\langle H, T\rangle \not\models \Gamma$. The term equilibrium model was coined in [30] and there is complete agreement between equilibrium models and the stable models of logic programs as defined in [31].

In an attempt to provide a solid, logical foundation for hybrid systems such as the ones introduced in Section 2.1, a number of extensions of HT for incorporating constraints have been introduced.

The *Logic of Here-and-There with constraints* (HT_c; [15]) allows for capturing constraint theories in the non-monotonic setting and has subsequently been extended with aggregate functions over constraint values and variables [32, 33]. In [33], specifications for aggregate functions in terms of HT_c are given based on two different semantic principles. While the semantics given in [34] ensures that aggregate terms are always defined, [35] prohibits so-called *vicious cycles*. We also refer to the former as *Ferraris* and to the latter as *Gelfond-Zhang* (GZ) aggregate semantics.

The *Bound-Founded Logic of Here-and-There* (HT_b; [36]) generalizes the concept of *foundedness* to integer and other types of variables. Intuitively, the idea is that variables get assigned the smallest value that can be justified in the program. This can be seen as a generalization of plain HT if one regards Boolean truth values as ordered by letting *true* be greater than *false*. Consider for example the fact consisting of the constraint $x \geq 7$ with x an integer variable. While in HT_c this has an infinite number of stable models $\{x \mapsto 7\}, \{x \mapsto 8\}, \{x \mapsto 9\}, \ldots$, only the first one would be a stable model in HT_b, assuming that the integers are ordered under the usual $\leq$ order.

Both of these extensions can be seen as *black-box* approaches in the sense that the constraints are incorporated as special entities whose syntax and satisfaction relations are generally left open. Thus, the intricacies of the hybrid part are mostly unknown from the logic program perspective. Another HT extension with a *white-box* approach of constraints is $ASP(\mathcal{AC})$ [37] which generalizes logical connectives as a particular case of more general operations on weighted formulas over semirings. In this setting, operators like logical conjunction $\wedge$ become just one more possible operation that can be combined with others, such as addition or multiplication (depending on the underlying semiring). This results in a very expressive and powerful formalism but at the price of a more complex semantics and the requirement of a semiring structure.

Further white-box approaches are based on the incorporation of intensional or non-Herbrand functions in ASP. For instance, [38] added partial intensional functions to a quantified First-Order version of HT [39] and later extended this to sets and aggregates [40].

2.3. Configuration

A wide range of approaches exist for representing and solving configuration problems across various paradigms [41, 42]. In recent years, ASP has emerged as a promising alternative, as evidenced by several applications [8, 43, 9, 44]. Moreover, [45] developed OOASP, an object-oriented approach to configuration using UML-style diagrams and which directly defines concepts in ASP. In the context of interactive configuration, [46] conducted a comparative evaluation of various systems for their suitability in this context. This included the ASP solver *clingo* as well as SAT and CP systems and assessed *clingo* to be as capable as any other system. An interactive configuratior for OOASP has been

[6] This is based on Kripke semantics for intuitionistic logic, see [29].

developed in [47] and extended in [48]. Another approach using the IDP-Z3 system has been developed in [49] and applied, for instance, to the configuration of adhesives [50].

2.4. Other approaches to grounding

Note that the literature also contains a range of alternative grounding approaches such as lazy grounding [51, 52], compilation of programs as propagators [53, 54], and body-decoupled grounding [55, 56]. The aim of these approaches is overcoming the aforementioned grounding bottleneck. However, while they have possible applications to the area of configuration [57], they do not form part of my active research. The reason behind this is twofold. First, my research is focused on the theoretical foundations of hybrid solvers. Second, many real-world problems require external inference methods that go beyond the current ASP-Core-2 [58] input language and thus beyond the formalisms these approaches are currently targeting.

3. Research

My research focuses on the foundations of hybrid ASP. The goal of this research is both to better understand its fundamental properties and to exploit this knowledge to guide and improve solver implementations. Furthermore, as this research is motivated by problems in real-world applications, another goal is to better understand the essence of these problems and how they can be solved using hybrid ASP. In other words, the objective is to find mathematical or logical formalizations of the problems, which subsequently serve as basis for succint but general ASP representations. These two goals are reciprocally beneficial as a deeper insight into real-world problems will make clearer the necessary research directions on the foundational level. In the context of applications, my current focus lies on problems in the realm of (industrial product) configuration.

3.1. Current Status of the Research

Regarding the theoretical aspects of my research, I recently found results [59] on the formal background of solver *clingo*[DL]. Here, we used the logic HT_b to provide a semantics for a subset of the syntax of *clingo*[DL] and to propose a new semantics for the fragment we called *logic programs with difference constraints*. Our semantics shows that the concept of assigning a minimal, founded value to integer variables as in HT_b matches with the way difference constraints in *clingo*[DL] are treated. In general, a system of difference constraints has multiple valid solutions but only a single minimal/canonical one.

The provided semantics of [59] is valid for the subset of programs that contain propositional as well as binary difference constraint atoms of the form $\&\texttt{diff}\{x - y\} \texttt{<=} d$, where x, y are integer variables and $d \in \mathbb{Z}$. The full syntax of *clingo*[DL] also allows for unary constraints and the goal is to extend our semantics to include them.

Furthermore, we showed in [59] that the logic HT_c can be embedded in HT_b. Since characterizations of *clingcon* and *flingo* exist for HT_c, this implies that HT_b can serve as formal foundation for these systems as well. We touch upon this topic briefly in the paper by introducing a basic unifying framework for the three systems in the restricted context of logic programs with difference constraints.

Future work on HT_b includes finding an operator-based characterization.

Another ongoing work regards the theoretical foundations of solver *flingo*. Here, one of the open issues is that current results in HT_c only allow for the use of GZ aggregate semantics in *flingo*. However, we would like to be able to use Ferraris aggregate semantics which guarantee definedness as known from *clingo*. Our current approach here consists of finding and proving a suitable translation between the two semantics.

On the practical side of my research, preliminary results have been found in application of (plain and hybrid) ASP to configuration problems. In [60], we developed a principled approach to configuration that included a mathematical formalization of configuration problems with an ASP-based solution which has been refined in [61]. We defined a configuration problem in terms of an abstract model

and a concrete instantiation. While the model serves as a blueprint for all possible configurations, the instantation represents a solution.

A similar but slightly different work has been done in [62] where we developed the CoomSuite [63], a workbench for experimentation with industrial-scale product configuration problems. The CoomSuite is built around product configuration language Coom[64][7] and provides a Coom grammar for parsing, a specialized ASP translator for conversion into facts, two encodings (one for *clingo* and one for *flingo*) as well as various benchmark sets, and a simple UI. The intention is to ease the development of powerful methods able to perform in industrial settings.

Future work here includes the further study of suitable representations for hybrid solver *flingo*. The current *flingo* encodings do not necessarily use all features the solver has to offer, e.g., undefinedness of numeric variables, but rather leaves this to non-hybrid ASP. The reason for this is that these encodings have been constructed with a plain ASP encoding as base, only modifying the necessary parts. An approach we want to pursue here is to find a logical formalization of configuration problems in terms of HT_c and use this as basic for new encodings which make more natural use of *flingo*'s features. We expect that this will not only improve the knowledge representation but also the performance of the solver.

3.2. Open issues and expected achievements

Proving the translation between GZ- and Ferraris aggregate semantics has so far been more difficult than expected.

The extension of the HT_b-semantics for *clingo*[DL] to cover unary constraints should be feasible, although, different approaches are possible and it is not yet obvious which one is better suited or whether they might even be equivalent.

Another open question for which no clear direction exists yet, is whether an efficient implementation for the newly proposed semantics exists. We would like to make first experiments in this direction by using meta-programming [18] in *clingo*.

Declaration on Generative AI

During the preparation of this work, the author used Gemini and GPT 4.1 in order to: Abstract drafting, Paraphrase and Reword. After using these tools, the author reviewed and edited the content as needed and takes 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] E. Erdem, M. Gelfond, N. Leone, Applications of ASP, AI Magazine 37 (2016) 53–68. doi:10.1609/aimag.v37i3.2678.

[3] A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. Teppan, Industrial applications of answer set programming, Künstliche Intelligenz 32 (2018) 165–176. doi:10.1007/s13218-018-0548-6.

[4] C. Dodaro, P. Gasteiger, N. Leone, B. Musitsch, F. Ricca, K. Schekotihin, Combining answer set programming and domain heuristics for solving hard industrial problems, Theory and Practice of Logic Programming 16 (2016) 653–669.

[5] O. Altay-Kern, E. Dietz, I. Kuhlmann, M. Thimm, Exploring desirable configurations in global logistics with heuristicsearch in answer set programming, in: M. Ortiz, R. Wassermann, T. Schaub (Eds.), Proceedings of the Twenty-second International Conference on Principles of Knowledge Representation and Reasoning (KR'25), 2025, pp. 578–587. doi:10.24963/KR.2025/56.

[7] Coom is a domain-specific language developed by denkbares GmbH and used in numerous industrial applications

[6] 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.

[7] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), Knowledge-Based Configuration: From Research to Business Cases, Elsevier/Morgan Kaufmann, 2014. doi:10.1016/C2011-0-69705-4.

[8] M. Gebser, R. Kaminski, T. Schaub, aspcud: A Linux package configuration tool based on answer set programming, in: C. Drescher, I. Lynce, R. Treinen (Eds.), Proceedings of the Second International Workshop on Logics for Component Configuration (LoCoCo'11), volume 65 of *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, 2011, pp. 12–25. doi:10.4204/eptcs.65.2.

[9] E. Gençay, P. Schüller, E. Erdem, Applications of non-monotonic reasoning to automotive product configuration using answer set programming, Journal of Intelligent Manufacturing 30 (2019) 1407–1422. doi:10.1007/s10845-017-1333-3.

[10] M. Banbara, B. Kaufmann, M. Ostrowski, T. Schaub, Clingcon: The next generation, Theory and Practice of Logic Programming 17 (2017) 408–461. doi:10.1017/S1471068417000138.

[11] T. Janhunen, R. Kaminski, M. Ostrowski, S. Schellhorn, P. Wanko, T. Schaub, Clingo goes linear constraints over reals and integers, Theory and Practice of Logic Programming 17 (2017) 872–888. doi:10.1017/S1471068417000242.

[12] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti, P. Wanko, Train scheduling with hybrid ASP, Theory and Practice of Logic Programming 21 (2021) 317–347. doi:10.1017/S1471068420000046.

[13] D. Rajaratnam, T. Schaub, P. Wanko, K. Chen, S. Liu, T. Son, Solving an industrial-scale warehouse delivery problem with answer set programming modulo difference constraints, Algorithms 16 (2023). URL: https://www.mdpi.com/1999-4893/16/4/216. doi:10.3390/a16040216.

[14] M. M. S. El-Kholany, M. Gebser, K. Schekotihin, Decomposition strategies and multi-shot ASP solving for job-shop scheduling, Logical Methods in Computer Science 21 (2025). doi:10.46298/LMCS-21(3:16)2025.

[15] P. Cabalar, R. Kaminski, M. Ostrowski, T. Schaub, An ASP semantics for default reasoning with constraints, in: S. Kambhampati (Ed.), Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16), IJCAI/AAAI Press, 2016, pp. 1015–1021. doi:10.5555/3060621.3060762.

[16] A. Heyting, Die formalen Regeln der intuitionistischen Logik, in: Sitzungsberichte der Preussischen Akademie der Wissenschaften, Deutsche Akademie der Wissenschaften zu Berlin, 1930, pp. 42–56.

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

[18] 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.

[19] A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2009. doi:10.3233/FAIA336.

[20] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, Strong equivalence in answer set programming with constraints, CoRR abs/2502.04302 (2025). URL: https://doi.org/10.48550/arXiv.2502.04302. doi:10.48550/ARXIV.2502.04302.

[21] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, On the semantics of hybrid ASP systems based on clingo, Algorithms 16 (2023). URL: https://www.mdpi.com/1999-4893/16/4/185. doi:10.3390/a16040185.

[22] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, `Flingo` — Instilling ASP expressiveness into numerical constraints, Theory and Practice of Logic Programming (2026) (to appear).

[23] C. Redl, The dlvhex system for knowledge representation: recent advances (system description), Theory and Practice of Logic Programming 16 (2016) 866–883. doi:10.1017/S1471068416000211.

[24] T. Janhunen, I. Niemelä, M. Sevalnev, Computing stable models via reductions to difference logic, 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. 142–154.

[25] G. Liu, T. Janhunen, I. Niemelä, Answer set programming via mixed integer programming, in: G. Brewka, T. Eiter, S. McIlraith (Eds.), Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12), AAAI Press, 2012, pp. 32–42.

[26] Y. Lierler, B. Susman, SMT-based constraint answer set solver EZSMT (system description), 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. 1:1–1:15. doi:10.4230/OASICS.ICLP.2016.1.

[27] T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, T. Kaminski, ASP-FZN: A translation-based constraint answer set solver, Theory and Practice of Logic Programming 25 (2025) 649–667. doi:10.1017/S1471068425100264.

[28] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Transactions on Computational Logic 2 (2001) 526–541. doi:10.1145/383779.383783.

[29] D. van Dalen, Intuitionistic logic, in: D. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic, volume 3, Springer-Verlag, 2001, pp. 225–339. doi:10.1007/978-94-009-5203-4_4.

[30] D. Pearce, A new logical characterisation of stable models and answer sets, in: J. Dix, L. Pereira, T. Przymusinski (Eds.), Proceedings of the Sixth International Workshop on Non-Monotonic Extensions of Logic Programming (NMELP'96), volume 1216 of *Lecture Notes in Computer Science*, Springer-Verlag, 1997, pp. 57–70. doi:10.1007/BFb0023801.

[31] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, K. Bowen (Eds.), Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88), MIT Press, 1988, pp. 1070–1080. doi:10.1201/b10397-6.

[32] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, An ASP semantics for constraints involving conditional aggregates, 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. 664–671. doi:10.3233/FAIA200152.

[33] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, A uniform treatment of aggregates and constraints in hybrid ASP, in: D. Calvanese, E. Erdem, M. Thielscher (Eds.), Proceedings of the Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR'20), AAAI Press, 2020, pp. 193–202. doi:10.24963/KR.2020/20.

[34] P. Ferraris, Logic programs with propositional connectives and aggregates, ACM Transactions on Computational Logic 12 (2011) 25:1–25:40. doi:10.1145/1970398.1970401.

[35] M. Gelfond, Y. Zhang, Vicious circle principle and logic programs with aggregates, Theory and Practice of Logic Programming 14 (2014) 587–601. doi:10.1017/S1471068414000222.

[36] P. Cabalar, J. Fandinno, T. Schaub, S. Schellhorn, Lower bound founded logic of here-and-there, in: F. Calimeri, N. Leone, M. Manna (Eds.), Proceedings of the Sixteenth European Conference on Logics in Artificial Intelligence (JELIA'19), volume 11468 of *Lecture Notes in Computer Science*, Springer-Verlag, 2019, pp. 509–525. doi:10.1007/978-3-030-19570-0_34.

[37] T. Eiter, R. Kiesel, Asp($\dashv\rfloor$): Answer set programming with algebraic constraints, Theory and Practice of Logic Programming 20 (2020) 895–910. doi:10.1017/S1471068420000393.

[38] P. Cabalar, Functional answer set programming, Theory and Practice of Logic Programming 11 (2011) 203–233. doi:10.1017/S1471068410000517.

[39] D. Pearce, A. Valverde, Towards a first order equilibrium logic for nonmonotonic reasoning, in: J. Alferes, J. Leite (Eds.), Proceedings of the Ninth European Conference on Logics in Artificial Intelligence (JELIA'04), volume 3229 of *Lecture Notes in Computer Science*, Springer-Verlag, 2004, pp. 147–160. doi:10.1007/978-3-540-30227-8_15.

[40] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, D. Pearce, Functional ASP with intensional sets: Application to Gelfond-Zhang aggregates, Theory and Practice of Logic Programming 18 (2018) 390–405. doi:10.1017/S1471068418000169.

[41] U. Junker, Configuration, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier Science, 2006, pp. 837–873. doi:10.1016/s1574-6526(06)80028-3.

[42] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, K. Wolter, Configuration knowledge

representation and reasoning, in: [7], 2014, pp. 41–72. doi:10.1016/b978-0-12-415817-7.00006-2.

[43] A. Felfernig, A. Falkner, M. Atas, S. Erdeniz, C. Uran, P. Azzoni, ASP-based knowledge representations for IoT configuration scenarios, in: L. Zhang, A. Haag (Eds.), Proceedings of the Nineteenth International Configuration Workshop (CONF'17), 2017, pp. 62–67.

[44] K. Herud, J. Baumeister, O. Sabuncu, T. Schaub, Conflict handling in product configuration using answer set programming, in: Proceedings of the Fifteenth Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'22), volume 3193 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3193/paper2ASPOCP.pdf.

[45] A. Falkner, A. Ryabokon, G. Schenner, K. Shchekotykhin, OOASP: connecting object-oriented and logic programming, in: F. Calimeri, G. Ianni, M. Truszczyński (Eds.), Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), volume 9345 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2015, pp. 332–345. doi:10.1007/978-3-319-23264-5_28.

[46] A. Falkner, A. Haselböck, G. Krames, G. Schenner, H. Schreiner, R. Taupe, Solver requirements for interactive configuration., Journal of Universal Computer Science 26 (2020) 343–373. doi:10.3897/jucs.2020.019.

[47] R. Comploi-Taupe, A. Falkner, S. Hahn, T. Schaub, G. Schenner, Interactive configuration with ASP multi-shot solving, in: [65], 2023, pp. 95–103. URL: https://ceur-ws.org/Vol-3509/paper13.pdf.

[48] L. Balázová, R. Comploi-Taupe, S. Hahn, N. Rühling, G. Schenner, Smart expansion techniques for ASP-based interactive configuration, Theory and Practice of Logic Programming 25 (2025) 473–488. doi:10.1017/S147106842510029X.

[49] P. Carbonnelle, S. Vandevelde, J. Vennekens, M. Denecker, Interactive configurator with FO(.) and IDP-Z3, CoRR abs/2202.00343 (2023). URL: https://arxiv.org/abs/2202.00343.

[50] S. Vandevelde, J. Vennekens, J. Jordens, B. Van Doninck, M. Witters, Knowledge-based support for adhesive selection: Will it stick?, tplp 24 (2024) 560–580. doi:10.1017/S1471068424000024.

[51] A. Weinzierl, R. Taupe, G. Friedrich, Advancing lazy-grounding ASP solving techniques — restarts, phase saving, heuristics, and more, Theory and Practice of Logic Programming 20 (2020) 609–624. URL: http://arxiv.org/abs/2008.03526.

[52] A. Dal Palù, A. Dovier, E. Pontelli, G. Rossi, GASP: Answer set programming with lazy grounding, Fundamenta Informaticae 96 (2009) 297–322.

[53] B. Cuteri, C. Dodaro, F. Ricca, P. Schüller, Overcoming the grounding bottleneck due to constraints in ASP solving: Constraints become propagators, in: C. Bessiere (Ed.), Proceedings of the Twenty-ninth International Joint Conference on Artificial Intelligence (IJCAI'20), ijcai.org, 2020, pp. 1688–1694.

[54] C. Dodaro, G. Mazzotta, F. Ricca, Blending grounding and compilation for efficient ASP solving, in: P. Marquis, M. Ortiz, M. Pagnucco (Eds.), Proceedings of the Twenty-first International Conference on Principles of Knowledge Representation and Reasoning (KR'24), 2024, pp. 317–328. doi:10.24963/KR.2024/30.

[55] V. Besin, M. Hecher, S. Woltran, Body-decoupled grounding via solving: A novel approach on the ASP bottleneck, in: L. De Raedt (Ed.), Proceedings of the Thirty-first International Joint Conference on Artificial Intelligence (IJCAI'22), ijcai.org, 2022, pp. 2546–2552. doi:10.24963/IJCAI.2022/353.

[56] A. Beiser, S. Woltran, M. Hecher, Automated hybrid grounding using structural and data-driven heuristics, Theory and Practice of Logic Programming 25 (2025) 489–506. doi:10.1017/S1471068425100173.

[57] V. Semmelrock, G. Friedrich, Investigating the grounding bottleneck for a large-scale configuration problem: Existing tools and constraint-aware guessing, in: M. Gebser, D. Inclezan, F. Ricca, M. Carro, M. Truszczynski (Eds.), Technical Communications of the Fourty-first International Conference on Logic Programming (ICLP'25), EPTCS, 2025, pp. 482–495. doi:10.4204/EPTCS.439.33.

[58] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, ASP-Core-2: Input language format, 2012. URL: https://www.mat.unical.it/aspcomp2013/ASPStandardization.

[59] P. Cabalar, J. Fandinno, N. Rühling, T. Schaub, S. Schellhorn, P. Wanko, Bound-founded semantics for answer set programming with difference constraints, EPTCS, 2026. (to appear).

[60] N. Rühling, T. Schaub, T. Stolzmann, Towards a formalization of configuration problems for ASP-based reasoning: Preliminary report, in: [65], 2023, pp. 85–94. URL: https://ceur-ws.org/Vol-3509/paper12.pdf.

[61] N. Rühling, T. Schaub, P. Wanko, Formalizing ASP-based product configuration: Work in progress, in: T. Geibinger, T. Stolzmann, A. Tarzariol (Eds.), Proceedings of the Eigth Workshop on Trends and Applications of Answer Set Programming (TAASP'25), 2025. URL: https://taasp.at/2025/assets/paper/1_Formalizing_ASP_based_Produc.pdf.

[62] J. Baumeister, S. Hahn, K. Herud, M. Ostrowski, J. Reutelshoefer, N. Rühling, T. Schaub, P. Wanko, Towards industrial-scale product configuration, Theory and Practice of Logic Programming (2026) 1–31. doi:10.1017/S1471068426100404.

[63] J. Baumeister, S. Hahn, J. Heuer, K. Herud, M. Ostrowski, J. Reutelshöfer, N. Rühling, T. Schaub, P. Wanko, CoomSuite, https://github.com/potassco/coom-suite, 2024.

[64] J. Baumeister, K. Herud, J. Reutelshöfer, V. Belli, Coom language, https://www.coom-lang.org/, 2024.

[65] J. Horcas, J. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proceedings of the Twenty-fifth International Configuration Workshop (CONF'23), volume 3509, CEUR Workshop Proceedings, 2023.