

Integrating SMT solvers into Goal-Directed Answer Set Programming, Challenges and Directions

Sarat Chandra Varanasi¹, Baoluo Meng¹

¹GE Aerospace Research, 1 Research Circle, Niskayuna, NY, USA

Abstract

Goal-Directed Answer Programming is a powerful paradigm to evaluate Answer Set Programs with several applications in Natural Language Understanding based question-answering, legal reasoning, development of LLM-assisted chatbots and reasoning about cyber-physical systems represented in the event-calculus formalism. Particularly, goal-directed ASP has been extended to support constraint-solving over reals CLP(R) to facilitate the capture of event calculus axioms. This enhancement involved integrating the CLP(R) constraint-solving system with the operational semantics of the goal-directed s(CASP) system. We position that more expressive and efficient theory-solving offered by modern SMT solvers can be integrated into s(CASP) to naturally evolve and increase the modeling capabilities of s(CASP). We show some potential applications of using a few SMT-based theories and also discuss some challenges involved in integrating SMT-solvers with s(CASP).

Keywords

Goal-Directed Answer Set Programming, SMT solving, Theory solving

1. Introduction

Goal-directed Answer Set Programming is a relatively newer paradigm for evaluating Answer Set Programs [1]. Traditionally Answer Set Programs have been evaluated using a declarative semantics in terms of computing *stable* models that satisfy a certain answer set program [2]. While evaluating answer set programs using grounding and satisfiability solving is extremely fast, goal-directed ASP complements traditional declarative ASP solving by providing a query driven approach to ASP program evaluation. The resulting effect is that ASP can be used similar to an interactive Prolog system while remaining consistent with the semantics of Answer Set Programs. The first goal-directed answer set solver was the *Galliwasp* system that defined the operational semantics for goal-directed ASP for propositional Answer Set Programs [3]. It crucially relies on the principle of co-induction and co-inductive logic programming for aligning with declarative ASP semantics. Later, goal-directed ASP solving for predicate answer set programs was developed by defining newer notions such as *variable disunification* to allow for a consistent operational semantics for the *negation-as-failure* operator in ASP [1]. The developed s(ASP) system supported predicate ASP with no support for constraint-solving present in traditional logic programming augmented with CLP(R) and CLP(FD) libraries. In 2017, support for constraint-solving over reals was integrated with the operational semantics of s(ASP) which


4th Workshop on Goal-directed Execution of Answer Set Programs (GDE'24), October 12, 2024

✉ saratchandra.varanasi@ge.com (S. C. Varanasi); baoluo.meng@ge.com (B. Meng)

🆔 0000-0002-4620-4266 (S. C. Varanasi); 0000-0002-3284-1969 (B. Meng)



© 2023 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)

resulted in the s(CASP) system [4]. s(CASP) ability to support CLP(R) has enabled the capture of axioms of Event calculus [5] and unlocked several applications in validation of systems requirements in different domains from Avionics [6] to Healthcare applications [7]. Research in formal methods using SAT-solving evolved into support for multiple expressive theories that has resulted in numerous applications in fields of Program Verification [8], Security Analyses [9], SMT-based model checking to mention a few [10]. Several industrial scale SMT solvers such as Z3 and CVC5 are continually extended to support more expressive theories with novel applications. For example, SMT solving over a restricted theory of strings has been applied to formally verify access policies in Amazon Web Services [11]. We position that integrating expressive theory solvers powered by advances in industrial scale SMT solver engines with s(CASP) will vastly improve in the accessibility of Formal Methods along and also drive s(CASP) towards the direction of being adopted as an SMT-enabled theorem-prover. The rest of the paper is organized as follows: We provide a background of SMT theories integrated into Declarative ASP followed by how CLP(R) was integrated into the operational semantics of goal-directed ASP. We then motivate the integration of SMT-solvers into goal-directed ASP with an example. Then, we present the challenges of integrating solvers into s(CASP) with potential future directions.

2. Background

2.1. Declarative ASP with constraint-solving

Integrating constraint-solving into declarative Answer Set Programming has been continually evolving from its introduction [12] to direct integration of SMT-solving capabilities in the Clingcon system [13]. The work of *Lierler* broadly categorizes declarative constraint-based ASP into two categories: *Integrational* where ASP solving is combined algorithmically with theory-solving and *Translational* where ASP programs are translated into SMT-LIB format [14]. The reader is referred to their work for a comprehensive survey of declarative constraint answer set programming (CASP) systems [14]. Our focus for this paper is solely on an integrational approach for SMT into s(CASP). For the integrational approaches the main challenge involves correctly and efficiently grounding the input ASP programming while also dispatching theory-atoms to the respective constraint solver such as an SMT-solver. Despite these challenges, such approaches have resulted in numerous applications and various implementations of CASP [14].

2.2. Integration of CLP(R) into s(ASP)

The operational semantics of first predicate ASP system s(ASP) introduced unique techniques to be consistent with ASP semantics:

1. Disunification operator between terms which maintains prohibited list of ground terms for each variable while performing top-down execution of a query
2. Dual rules to provide operational semantics for negation-as-failure (NAF) predicates present in the original ASP program
3. Forall algorithm to evaluate the success of a goal for all possible values of its variables, resulting from dual rules generated for original ASP program rules with existentially quantified variables in the body.

More details about s(ASP) can be found elsewhere [1]. In order to modularly introduce CLP(R) into s(ASP), the disunification operator was extended to also represent CLP(R) constraints for relevant terms and the *forall* was extended to check for a CLP(R) constrained variable for goal validity for its complete range. More details about the resulting s(CASP) system can be found elsewhere [4].

3. Example of potential SMT theory integrated into s(CASP)

s(CASP) currently supports linear arithmetic over reals. While this has proved to be useful, there are aerospace applications such as that require solving non-linear constraints such as aircraft collision detection [15]. Consider an object A's trajectory confined to the 2-D unit circle whereas another object B's trajectory confined to the 2-D half-plane where its x-coordinate is greater than 1. Clearly, we know that the object A's space does not intersect with that of object B, but we cannot simplify it currently in s(CASP).

```
intersect :- areaA(X, Y), areaB(X). % intersect cannot be satisfie
areaA(X, Y) :- X*X + Y*Y #< 1. % non-linear arithmetic constraint
areaB(X) :- X #> 1.
```

There are two aspects of integrating SMT-solver into s(CASP) beyond the obvious aspect of increased expressivity of theories.

1. Removing SMT-LIB encoding overhead: Variables and terms naturally express theory constraints in Logic Programming and appropriately implementing the SMT-LIB encoding, solver dispatch and integration of c-*forall* for the respective theory readily adds theory-solving into s(CASP).
2. Such an integration drives the use of s(CASP) as a front-end to decision procedures. Although, removing SMT-LIB encoding overhead implies having developed a front-end, modular integration of theory-solving into s(CASP) goes a step beyond dispatching SMT queries. Proof obligations pertaining to an application domain can be structured into SMT-LIB solver dispatches to establish formal proofs of correctness or produce counter-examples for system functions.

4. Challenges and Directions

Although the above example motivated the need for more decision procedures powered by SMT technology to be integrated into s(CASP), there are a few challenges:

1. Before a query is issued to s(CASP), static analysis can help infer the right SMT theory that needs to be used before dispatching SMT-encoding and solving. This challenge can be circumvented by modularizing rules according to the SMT theory solving they would require.
2. Non-termination issues: s(CASP) suffers from non-termination when fresh variables are added to its co-inductive hypothesis set that builds an infinite (descending) ascending chain of variable inequalities. We term this the "Zeno" problem. This problem is still open for s(CASP).

In addition to the challenges, other directions for s(CASP)-SMT integration would be to develop dynamic consistency checking methods [16] while accounting for SMT-LIB theories. Another direction is to integrate proof visualizers developed for SMT solvers such as CVC5 [17] for SMT unsatisfiable queries with the justification tree that is already produced by s(CASP).

References

- [1] K. Marple, E. Salazar, G. Gupta, Computing stable models of normal logic programs without grounding, preprint arXiv:1709.00501 (2017).
- [2] M. Gebser, R. Kaminski, B. Kaufmann, P. Lühne, s. Obermeier, M. Ostrowski, J. Romero, T. Schaub, S. Schellhorn, P. Wanko, The potsdam answer set solving collection 5.0, *KI-Künstliche Intelligenz* 32 (2018) 181–182.
- [3] K. Marple, G. Gupta, Galliwasp: A goal-directed answer set solver, in: *International Symposium on Logic-Based Program Synthesis and Transformation*, Springer, 2012, pp. 122–136.
- [4] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, *Theory and Practice of Logic Programming* 18 (2018) 337–354.
- [5] J. Arias, M. Carro, Z. Chen, G. Gupta, Modeling and reasoning in event calculus using goal-directed constraint answer set programming, *Theory and Practice of Logic Programming* 22 (2022) 51–80.
- [6] B. Hall, S. C. Varanasi, J. Fiedor, J. Arias, K. Basu, F. Li, D. Bhatt, K. Driscoll, E. Salazar, G. Gupta, Knowledge-assisted reasoning of model-augmented system requirements with event calculus and goal-directed answer set programming, arXiv preprint arXiv:2109.04634 (2021).
- [7] O. Vašíček, J. Arias, J. Fiedor, G. Gupta, B. Hall, B. Křena, B. Larson, S. C. Varanasi, T. Vojnar, Early validation of high-level system requirements with event calculus and answer set programming, arXiv preprint arXiv:2408.09909 (2024).
- [8] N. Bjørner, L. de Moura, Applications of smt solvers to program verification, *Notes for the Summer School on Formal Techniques* (2014).
- [9] P. B. Jackson, B. J. Ellis, K. Sharp, Using smt solvers to verify high-integrity programs, in: *Proceedings of the second workshop on Automated formal methods*, 2007, pp. 60–68.
- [10] C. Tinelli, Smt-based model checking., in: *NASA Formal Methods*, 2012, p. 1.
- [11] J. Backes, P. Bolignano, B. Cook, C. Dodge, A. Gacek, K. Luckow, N. Rungta, O. Tkachuk, C. Varming, Semantic-based automated reasoning for aws access policies using smt, in: *2018 Formal Methods in Computer Aided Design (FMCAD)*, IEEE, 2018, pp. 1–9.
- [12] V. S. Mellarkod, M. Gelfond, Y. Zhang, Integrating answer set programming and constraint logic programming, *Annals of Mathematics and Artificial Intelligence* 53 (2008) 251–287.
- [13] M. Banbara, B. Kaufmann, M. Ostrowski, T. Schaub, Clingcon: The next generation, *Theory and Practice of Logic Programming* 17 (2017) 408–461.
- [14] Y. Lierler, Constraint answer set programming: Integrational and translational (or smt-based) approaches, *Theory and Practice of Logic Programming* 23 (2023) 195–225.
- [15] S. Paul, B. Meng, C. Alexander, Smt-based aircraft conflict detection and resolution, in: *NASA Formal Methods Symposium*, Springer, 2024, pp. 186–203.

- [16] J. Arias, M. Carro, G. Gupta, Towards dynamic consistency checking in goal-directed predicate answer set programming, in: International Symposium on Practical Aspects of Declarative Languages, Springer, 2022, pp. 117–134.
- [17] H. Barbosa, C. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, et al., cvc5: A versatile and industrial-strength smt solver, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2022, pp. 415–442.