KI - Künstliche Intelligenz (2018) 32:187–189
https://doi.org/10.1007/s13218-018-0535-y

SYSTEMS DESCRIPTION

The DLVHEX System

Thomas Eiter[1] · Stefano Germano[2] · Giovambattista Ianni[2] · Tobias Kaminski[1] · Christoph Redl[1] · Peter Schüller[1] · Antonius Weinzierl[1]

Published online: 15 May 2018

Abstract
HEX programs extend ASP with external atoms implemented in C++ or Python. DLVHEX is a solver for HEX that permits cyclic reasoning over external atoms and external value invention.

Keywords Answer set programming · External source access · Nonmonotonic reasoning

1 Motivation

Answer set programming (ASP) [1] is a logic programming formalism for knowledge representation and reasoning. However, in many applications, rules are insufficient for representing all reasoning about the domain of interest. Instead, accessing information or computations from the world outside the program is needed. HEX programs [5] extend ASP with special *external atoms* in rule bodies, whose truth value is determined by an external source using, e.g., imperative code.

Consider the following simple example:

$$critical(R) \leftarrow robot(R), \&battery[R](Lvl), \quad Lvl < 25.$$

where the status of some robots is determined by reading and interpreting their battery level. External oracles can perform arbitrary computations to determine their truth value. Two distinguishing features of HEX are:

- external computations can produce new constants,
- external computations may depend (possibly cyclically) on the true extension of predicates in a respective answer set.

These two features make HEX a widely applicable and also very expressive formalism.

2 HEX Formalism

HEX programs [5] extend ASP by allowing the use of *external atoms* of the form $\&g[p_1, \dots, p_k](c_1, \dots, c_l)$ in rule bodies, where $\&g$ is an *external predicate*, $p_1, \dots, p_k$ are input predicate names or constants, and $c_1, \dots, c_l$ are output terms. Intuitively, the semantics of an external predicate $\&g$ defines for a given assignment and input list $p_1, \dots, p_k$ for which lists of output constants $c_1, \dots, c_l$ the external atom should be true. The notions of rule and program satisfaction and of answer sets are then extended from ASP to HEX in the obvious way.

Importantly, external oracles can be true for output constants that do not occur in the input program to extend the

This research has been supported by the Austrian Science Fund (FWF) projects P27730 and W1255-N23.

✉ Tobias Kaminski
kaminski@kr.tuwien.ac.at

Thomas Eiter
eiter@kr.tuwien.ac.at

Stefano Germano
germano@mat.unical.it

Giovambattista Ianni
ianni@mat.unical.it

Christoph Redl
redl@kr.tuwien.ac.at

Peter Schüller
ps@kr.tuwien.ac.at

Antonius Weinzierl
aweinz@kr.tuwien.ac.at

[1] Institut für Logic and Computation, TU Wien, Favoritenstraße 9-11, 1040 Vienna, Austria

[2] Department of Mathematics and Computer Science, University of Calabria (UNICAL), Via Bucci, Cubo 30B, 87036 Rende, CS, Italy

domain of the original program (called *value invention*) [6]. This permits HEX programs to import knowledge from external sources, e.g., the Internet, as well as to reason with that knowledge. HEX is Σ_2^P-complete for ground programs if all external sources are polynomial and up to Turing-complete in general.

3 DLVHEX Solver

The DLVHEX solver[1] implements HEX programs based on Gringo and Clasp as backends and supports external atoms to be implemented in C++ and Python using an API provided by the reasoner. While grounding and ground program solving are often separated in two phases in ASP, in HEX the potential value invention in external sources requires them to be interleaved.

The basic approach [5] for evaluating ground HEX programs is based on a guess and check rewriting of the HEX program to normal ASP, where the checking part requires verification calls to the external sources. However, for scalability reasons, this approach has been extended by advanced evaluation techniques.

Related to DLVHEX are WASP's extension with external Python propagators [4], and CLINGO 5 [9], which provides generic interfaces for ASP modulo theory solving. While DLVHEX aims at user-friendly integration of heterogeneous sources, CLINGO 5 is geared to technically integrate specific theories, with limited value invention and possibly unfounded external recursion.

For getting started with HEX and the DLVHEX solver, a manual[2] and a step-by-step walkthrough for building an application based on HEX [7] are provided.

4 Applications

Angry-HEX. This AI agent plays the popular physics-based game *Angry Birds*[3] autonomously [2]. The aim of the game is to use a slingshot to shoot birds of different types at pigs placed on a scene in order to destroy them. Pigs are usually protected by obstacles of different types. The game uses a realistic physics simulation, including gravity and statics. The agent is publicly available as open-source software.[4] For a detailed explanation we refer to [2].

The reasoning part of the agent is implemented using logic programming. Plain ASP is insufficient as the computation involves physics simulation and floating point numbers to realize geometrical reasoning about properties, positions and relationships of the objects in the game. We used HEX programs, which allow for handling such computations by external atoms while the actual planning can be conveniently done by rules. For instance, the external atom $\&canpush[ao](A, B)$ finds all the pairs $\langle A, B\rangle$ of objects such that A can horizontally push B if it is hit by a bird or by another object. This computation requires geometric considerations about location and orientation of the objects.

The reasoning involves also the evaluation of the trajectory of the shot that will inflict the most useful direct and indirect damage to the target objects of the scene, which has been realized by using weak constraints.

Due to the generality of the approach, HEX has been successfully utilized in many further applications. At this, HEX is suited for importing information from external sources as well as outsourcing computations.

Besides simple plugins for querying *RDF* data sources or performing string operations (see the system website), one of the first elaborated use cases for HEX was an interface to a *Description Logics* (*DL*) reasoner, such that a DL ontology can be queried as well as manipulated from within a HEX program. Due to its modular architecture, HEX also makes it easy to plug in other external reasoners, e.g. a constraint solver, which has been exploited in an application that externally checks the satisfiability of arithmetic constraints used in a HEX program. Moreover, it is possible to leverage different types of external sources in the same HEX program.

The HEX formalism has also been deployed for solving more involved real-world problems, e.g. for planning in the area of robotics [8], for integration of biomedical databases [8], or for accessing data streams and classifying events with an DL ontology [3].

The mentioned applications, while cumbersome to formalize in many other formalisms, can be implemented in a natural way using HEX, which allows to combine very diverse modes of reasoning. Consequently, the continuous efforts directed towards making the HEX algorithm more efficient in general, as well as for particular classes of programs, pays off in terms of its increasing usefulness for solving practical problems.

A broader overview of HEX applications, discussion of problem solving and further references can be found in [7].

References

1. Schaub T, Woltran S (2018) Answer set programming unleashed! KI **(forthcoming)**
2. Calimeri F, Fink M, Germano S, Humenberger A, Ianni G, Redl C, Stepanova D, Tucci A, Wimmer A (2016) Angry-HEX: an artificial player for angry birds based on declarative knowledge bases. IEEE Trans Comput Intell AI Games 8(2):128–139

[1] www.kr.tuwien.ac.at/research/systems/dlvhex/.

[2] www.kr.tuwien.ac.at/research/systems/dlvhex/docs/userguide.pdf.

[3] https://www.angrybirds.com/games/angry-birds.

[4] https://github.com/DeMaCS-UNICAL/Angry-HEX.

3. Do TM, Loke SW, Liu F (2011) Answer set programming for stream reasoning. In: Proceedings of Canadian conference on artificial intelligence, pp 104–109
4. Cuteri B, Dodaro C, Ricca F, Schüller P (2017) Constraints, lazy constraints, or propagators in ASP solving: an empirical analysis. Theory Pract Logic Program 17(5–6):780–799
5. Eiter T, Fink M, Ianni G, Krennwallner T, Redl C, Schüller P (2016) A model building framework for answer set programming with external computations. Theory Pract Logic Program 16(04):418–464
6. Eiter T, Fink M, Krennwallner T, Redl C (2016) Domain expansion for ASP-programs with external sources. Artif Intell 233:84–121
7. Eiter T, Kaminski T, Redl C, Schüller P, Weinzierl A (2017) Answer set programming with external source access. Reason Web LNCS 10370:204–275
8. Erdem E, Gelfond M, Leone N (2016) Applications of answer set programming. AI Mag 37(3):53–68
9. Gebser M, Kaminski R, Kaufmann B, Ostrowski M, Schaub T, Wanko P (2016) Theory solving made easy with clingo 5. In: ICLP (Tech. Comm.), OASIcs, vol 52, pp 2:1–2:15