The IDP System

Johan Wittocx Broes De Cat Marc Denecker

Department of Computer Science, K.U. Leuven

Abstract

This paper presents the IDP system, a finite model generator for extended first-order logic theories. IDP can be used as a didactic tool in courses on (first-order) logic. It can also be applied to solve various constraint problems and for lightweight verification tasks.

1 Introduction

One of the long term research goals of the Knowledge Representation and Reasoning group at the K.U. Leuven is to build a *knowledge base system* (KBS) [4]. In such a system, knowledge about a domain of discourse is stored using a suitable logic. Next, different tasks are solved by applying various inference methods on that knowledge. An example is a KBS storing knowledge about course scheduling at a university. By applying suitable forms of inference, schedules can be generated automatically at the start of the year, hand-made schedules can be checked, existing schedules can be revised, etc., all using the same knowledge.

Our aim is to build a KBS with the logic FO($\cdot$) as knowledge representation language. FO($\cdot$) extends full first-order logic with useful constructs such as inductive definitions [3], aggregates and integer arithmetic.

The IDP system implements one form of inference for FO($\cdot$), namely finite model expansion. The task of model expansion is to expand a finite interpretation I for part σ of the vocabulary Σ of a given logic theory T to a model of T. Model expansion generalizes both model checking (if $\sigma = \Sigma$) and model generation for a given finite domain (if σ is empty).

The IDP system can currently be used to represent and solve problems in NP. It shares applications with other model generation systems such as Answer Set Programming (ASP) solvers [1], Constraint Programming (CP) systems (e.g., [9]), and the ALLOY system [2]. For instance, IDP can be used to solve scheduling, planning, and diagnosis problems in a purely declarative manner, it can be applied for lightweight software- and dynamic system verification, etc. One of the features that distinguishes IDP from ASP, CP and ALLOY is that it implements full first-order logic. This makes IDP also a suitable tool in courses on (first-order) logic (see below).

2 Technology

The IDP system consists of two main components:

- The *grounder* reduces an FO($\cdot$) theory to an equivalent propositional theory [11]. The grounder combines state-of-the-art grounding technology with a novel symbolic reasoning algorithm to derive which formulas of the propositional theory are certainly true or certainly false in models of that theory. This information is exploited to efficiently simplify the propositional theory while it is constructed.

- The *solver* searches for models of the propositional theory generated by the grounder. The core of the solver is a SAT solver, i.e., a model generator for propositional logic. Currently, we use the SAT solver MINISAT [7]. Propagation techniques from CP and SAT modulo theories (SMT) [10], as well as specialized algorithms [8], complement the SAT-based search to efficiently handle additional language constructs like aggregates and inductive definitions. The modular design, based on the SMT paradigm, allows for easy extension and combination with existing systems.

3 Current usage

IDP is used as an experimentation platform for students in several courses at the K.U.Leuven, among which the courses “First-order Logic”, “Knowledge representation” and “Modelling of Complex Systems”. Its application ranges from getting hands-on experience with abstract logical concepts to solving planning, scheduling, configuration and verification problems. The didactic tool LOGICPALET [6] uses IDP as a plug-in to automatically generate counterexamples when a student writes an incorrect logic translation of a statement in natural language.

There is ongoing development towards practical applications, for example in course and train (re)-scheduling and in building knowledge-driven user interfaces.

IDP participated in the second ASP competition, finishing fourth out of sixteen systems [5].

4 System

The IDP system is open-source and tested under Linux, Mac OS and Windows. It can be downloaded from `http://dtai.cs.kuleuven.be/krr/software`. The IDP system is being developed by Maarten Mariën, Johan Wittocx and Broes De Cat.

References

[1] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] Felix Chang. Alloy analyzer 4.0. `http://alloy.mit.edu/alloy4/`, 2007.

[3] Marc Denecker and Eugenia Ternovska. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL)*, 9(2):Article 14, 2008.

[4] Marc Denecker and Joost Vennekens. Building a knowledge base system for an integration of logic programming and classical logic. In María García de la Banda and Enrico Pontelli, editors, *ICLP*, volume 5366 of *LNCS*, pages 71–76. Springer, 2008.

[5] Marc Denecker, Joost Vennekens, Stephen Bond, Martin Gebser, and Mirosław Truszczyński. The second Answer Set Programming competition. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *LPNMR*, volume 5753 of *LNCS*, pages 637–654. Springer, 2009.

[6] Jan Denef. Logicpalet. `http://www.logicpalet.com`.

[7] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.

[8] Maarten Mariën. *Model Generation for ID-Logic*. PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, February 2009.

[9] Laurent Michel and Pascal Van Hentenryck. The Comet programming language and system. In Peter van Beek, editor, *CP*, volume 3709 of *LNCS*, pages 881–881. Springer, 2005.

[10] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

[11] Johan Wittocx, Maarten Mariën, and Marc Denecker. Grounding FO and FO(ID) with bounds. *Journal of Artificial Intelligence Research*, 38:223–269, 2010.