IPROVERPLAN: a system description

Konstantin Korovin
University of Manchester,
Manchester, UK

Martin Suda[*]
Vienna University of Technology,
Vienna, Austria

Introduction

IPROVERPLAN is an automated planning system that combines searching for plans and proving non-existence of solutions. In the Unsolvability International Planning Competition (Muise and Lipovetzky, 2016) only non-existence of solutions is reported.

The idea behind IPROVERPLAN is to lift a traditional encoding of planning into SAT (Kautz and Selman, 1992) to first-order level and to use an extension of the first-order theorem prover IPROVER (Korovin, 2008) for solving (non-)reachability questions in thus obtained first-order transition system. Thus the main feature of IPROVERPLAN is that it does not start the solving process by grounding the PDDL input. The planner also uses the lifted (i.e. first order) invariants produced by the algorithm adapted from Helmert (2009, Sect. 5) for pruning the search space.

First-Order Transition Systems

We encode planning domains as transition systems represented in (many-sorted) first-order logic. One of the main motivations behind this encoding is that first-order logic provides a higher level representation compared to propositional logic and in particular, avoids upfront grounding of the problem and at the same time reasoning can still be done efficiently by first-order theorem provers.

Our encoding falls into the effectively propositional (EPR) fragment of first-order logic which in the clausal form consists of sets of first-order clauses that do not contain function symbols other than constants. The EPR fragment is decidable (NEXPTIME-complete) and there are efficient calculi and systems for reasoning within this fragment.

We use EPR-based bounded model checking (BMC) for solving reachability problems (Emmer et al., 2012; Pérez and Voronkov, 2007; Emmer et al., 2010) and an extension of BMC with k-induction for solving non-reachability problems (Khasidashvili et al., 2015). In a nutshell, bounded model checking solves the reachability problem by symbolically unrolling the transition relation upto some bound n, and checking satisfiability of the resulting formula. If the BMC-unrolling of the system is satisfiable at a bound n then a goal state is reachable in n steps from an initial state and in this case we are done, otherwise we repeat unrolling of the system with an increased bound $n + 1$. One of the advantages of using a first-order encoding is that the system representation is not copied during the unrolling which can be the case with propositional translations.

The encoding

IPROVERPLAN lifts traditional encodings of planning into SAT (Kautz and Selman, 1992) to first-order level. This is straightforward in the sense that predicates which are usually introduced for a SAT encoding have naturally positions for arguments and our encoding supplies universal first-order variables for these arguments instead of exhaustively grounding the predicates. However, there are various subtleties connected with the lifting which need to be addressed.

Multiple types/sorts. PDDL benchmarks can declare and make use of a hierarchy of types, including disjunctive types. This hierarchy needs to be flattened in order to be mapped to many-sorted first order logic.

Finite domain. It is in general necessary to express that the domain of discourse contains only the declared objects and that these objects are distinct. This can be expressed with the help of the equality predicate, but, in particular, the distinctness criterion leads to quadratically many axioms in the number of objects.

Negative information about the initial state. Although the initial state is in PDDL described using only positive information about the facts that hold, a first order encoding needs to also negatively express all the facts which do not hold. We avoid generating all the "non-mentioned", potentially exponentially many ground facts, by using the equality predicate.

First-delete-then-add semantics. Even for problems officially declared as STRIPS, we sometimes need to resort to techniques for expressing conditionality of effects. That is because two first-order effects of an action may contradict each other and the semantics of PDDL dictates that in such a case the positive effect should have priority and be reflected in the successor state.[1]

[*]Supported by the ERC Starting Grant 2014 SYMCAR 639270 and the Austrian research project FWF RiSE S11409-N23.

[1]Without this extra measure the action would erroneously be-

Skolemization. At each time step of the modelled plan at least one action is applied. In the first-order lifting, we do not know which specific arguments will an action take. Thus these arguments are modelled as existential variables in the encoding and need to be Skolemized during translation to clause normal form. In order to stay within the EPR fragment, Skolem functions are translated into Skolem predicates (Baumgartner et al. (2009); Khasidashvili et al. (2015)).

There are two encodings we lifted into first order and experimented with. A serial encoding with an at-least-one axiom and classical frame axioms (McCarthy and Hayes, 1969) and a parallel encoding with mutual exclusions and explanatory frame axioms (Haas, 1987). We refer to (Ghallab et al., 2004, ch. 7.4) for further details. The competition version of iPROVERPLAN uses the serial encoding.

iProver

iProver is an automated theorem prover for many-sorted first-order logic, based on an instantiation calculus Inst-Gen (Korovin, 2013, 2008). The basic idea behind Inst-Gen is to interleave model-guided on demand instantiations of first-order formulae with propositional reasoning in an abstraction-refinement scheme. The calculus behind iProver is a decision procedure for the EPR fragment and iProver is particularly efficient in this fragment (Sutcliffe, 2014). iProver incorporates first-order bounded model checking and k-induction which we utilised for solving planning (un)-reachability problems.

iProver is implemented in OCaml and incorporates a wide range of simplification and preprocessing techniques (Korovin, 2008; Khasidashvili and Korovin, 2016). iProver uses MiniSAT (Eén and Sörensson, 2004) for reasoning with ground abstractions and Vampire for clausification (Kovács and Voronkov, 2013; Hoder et al., 2012).

The architecture

As a computer program, iPROVERPLAN consists of three main parts. The first part is a PDDL parser and encoder written in python. Given a PDDL input, it generates two outputs: 1) the encoded first-order transition system and 2) a Prolog representation of the input used by the invariant generator.

The second part is an SWI-Prolog implementation of the invariant generating algorithm described by Helmert (2009, Sect. 5). The invariants produced by this part enrich the transition system as universally quantified clauses referring to every time moment. Although logically redundant they enable early pruning of obviously unreachable states.

Finally, the transition system is translated into first-order conjunctive normal form by Vampire and passed to iProver.

References

P. Baumgartner, A. Fuchs, H. de Nivelle, and C. Tinelli. Computing finite models by reduction to function-free clause logic. *J. Applied Logic*, 7(1):58–74, 2009.

N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT'03*, pages 502–518. Springer, 2004.

M. Emmer, Z. Khasidashvili, K. Korovin, C. Sticksel, and A. Voronkov. EPR-based bounded model checking at word level. In *IJCAR*, pages 210–224, 2012.

Moshe Emmer, Zurab Khasidashvili, Konstantin Korovin, and Andrei Voronkov. Encoding industrial hardware verification problems into effectively propositional logic. In *FMCAD*, pages 137–144, 2010.

Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning – theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6.

Andrew R. Haas. The case for domain-specific frame axioms. In *The Frame Problem in Artificial Intelligence, Proceedings of the 1987 Workshop on Reasoning about Action*. Morgan Kaufmann, 1987.

Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artif. Intell.*, 173(5-6):503–535, 2009.

Krystof Hoder, Zurab Khasidashvili, Konstantin Korovin, and Andrei Voronkov. Preprocessing techniques for first-order clausification. In *Formal Methods in Computer-Aided Design, FMCAD*, pages 44–51, 2012.

Henry A. Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, pages 359–363, 1992.

Zurab Khasidashvili and Konstantin Korovin. Predicate elimination for preprocessing in first-order theorem proving. In *SAT'16*, page to appear, 2016.

Zurab Khasidashvili, Konstantin Korovin, and Dmitry Tsarkov. EPR-based k-induction with counterexample guided abstraction refinement. In *GCAI 2015*, pages 137–150. EasyChair, 2015.

Konstantin Korovin. iProver – an instantiation-based theorem prover for first-order logic (system description). In *IJCAR 2008*, volume 5195 of *LNCS*, pages 292–298. Springer, 2008.

Konstantin Korovin. Inst-Gen - a modular approach to instantiation-based automated reasoning. In *Programming Logics*, pages 239–270. Springer, 2013.

Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *CAV 2013*, pages 1–35, 2013.

John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

Christian Muise and Nir Lipovetzky. Unsolvability international planning competition. http://unsolve-ipc.eng.unimelb.edu.au/, February 2016.

Juan Antonio Navarro Pérez and Andrei Voronkov. Encodings of bounded LTL model checking in effectively propositional logic. In *CADE-21*, pages 346–361, 2007.

Geoff Sutcliffe. The CADE-24 automated theorem proving system competition - CASC-24. *AI Com.*, 27(4):405–416, 2014.

come unaplicable in the encoding due to the contradicting effects.