Property Directed Reachability for Automated Planning*

Martin Suda
Computer Science Department
University of Manchester, UK
martin.suda@gmail.com

Abstract

Property Directed Reachability (PDR), also known as IC3, is a very promising recent method for deciding reachability in symbolically represented transition systems. While originally conceived as a model checking algorithm for hardware circuits, it has already been successfully applied in several other areas. This paper summarizes the first investigation of PDR from the perspective of automated planning.

Introduction

Property Directed Reachability (PDR), also known as IC3, is a recently proposed algorithm for deciding reachability in symbolically represented transition systems (Bradley 2011; Eén, Mishchenko, and Brayton 2011). Since its discovery in 2010 it has already established itself as one of the strongest model checking algorithms used in hardware verification. The original way in which PDR harnesses the power of a modern SAT-solver gives the algorithm a unique ability to discover long counterexample paths combined with a remarkable performance in proving unreachability. Other interesting traits include a typically small memory footprint and a good potential for parallelization.

With awareness of the well-known equivalence between model checking and automated planning, the aim of this work is to investigate PDR from the planning perspective. Our main goal is to establish whether the practical success of the algorithm can be repeated on planning benchmarks. Moreover, we are also interested in the relation of PDR to the currently used planning techniques. This extended abstract is a summary of our findings, which have been presented in full elsewhere (Suda 2014).

Symbolic Transition Systems and Encodings

Transition systems are used to describe the state spaces and thus the semantics of problems both in model checking and in automated planning. We introduce the notion of a *symbolic transition system* (STS) to serve as a succinct representation of the state space. Based on propositional logic, an STS is designed to allow atomic queries about the system to be easily delegated to and answered by a SAT-solver. Symbolic transition systems serve as a general input to PDR.

In order to obtain an STS from a planning task we need to choose a suitable *encoding*. We observe that most of the standard encoding schemes of the planning as satisfiability paradigm (Kautz and Selman 1996) can be easily adopted for this purpose. This means that a general implementation of PDR combined with any such encoding already yields a standalone planner.

The Algorithm

PDR is probably best understood as a hybrid between an explicit and a symbolic approach. It explicitly constructs a path consisting of concrete states, starting from the initial state and extending it step by step towards the goal. At the same time, it maintains symbolic reachability information, which is locally refined whenever the current path cannot be extended further. The reachability information guides the path construction and is bound to eventually converge to a certificate of non-reachability, if no witnessing path exists.

In more detail, the reachability information takes the form of a sequence of sets of clauses. The i-th set in the sequence *over-approximates* the i-fold preimage of the set of goal states. These clause sets play a role similar to an admissible heuristic. They represent a lower bound estimate for the distance of a state to the goal and thus provide a means to guide the search towards it. However, while a heuristic value of a particular state is normally computed only once and it remains constant during the search for a plan, the clause sets in PDR are refined continually. The refinement happens on demand, driven by the states encountered during the search.

Since the path construction happens in the context of a concrete encoding, PDR can be likened to an instance of the planning as satisfiability approach in which the construction of the assignment is controlled to grow only in one direction. PDR also proceeds iteratively, gradually disproving existence of plans of length 0, 1, 2,.... Due to a so called *obligation rescheduling* technique, however, PDR can discover a plan of length l already during iteration $k < l$, that is, while the existence of shorter plans has not necessarily been ruled out yet. This typically leads to improved performance, as it allows the algorithm to avoid completing the potentially expensive non-existence proofs.

*The work was done in Max-Planck-Institut für Informatik, Saarbrücken, Germany.

PDR without a SAT-solver

Our main theoretical contribution is the discovery that in the context of automated planning PDR can be realized completely without a SAT-solver. We notice that the computation of the path *extension*, which is normally implemented by posing a query to the solver, can be in the case of the sequential plan semantics delegated to a planning-specific procedure. With this procedure we not only obtain a polynomial time guarantee for computing the extension, but by decoupling PDR from the underlying SAT-solver we also gain additional insights and ideas for further improvements.

The general form of the path extension query, the core operation around which the whole algorithm is built, is as follows. Given a state s and a set of clauses L, we ask whether there is a state t, a successor of s with respect to the encoded transition relation T, that satisfies the clauses of L. In the positive case, we take the successor t to proceed with the path construction. In the negative case, we want to extract a *reason* r in the form of a subset of the literals defining s such that no state satisfying r has a successor satisfying L. The reason r is then converted to a clause which refines the reachability information. Subsequently, PDR backtracks from s to consider a different state for extension.

When the encoded transition relation T follows the sequential plan semantics, the positive part of the above "contract" can be efficiently implemented by a simple iteration over all the successors of the state s. We just check for every action whether it is applicable in s and if so, we test whether the corresponding successor satisfies the clauses from L.

Non-trivial is, however, the negative part, because it is essential for the efficiency of PDR that the computed reason be as small as possible. We propose (Suda 2014) to decompose the problem by computing separate reasons for each action and combining these into an overall reason in a final optimization step. The complexity is polynomially bounded in the sizes of s, L, and the action set of the planning task.

Experiments

We performed a series of experiments aimed to establish the potential of PDR for automated planning. As the test problems we used 1561 STRIPS problems (of 49 domains, in total) collected from the International Planning Competition (IPC),[1] years 1998–2011. The experiments were run on machines with 3.16 GHz Intel Xeon CPU, 16 GB RAM, running Debian 6.0. Below we list the most interesting findings.

- The "SAT-solver free" variant of PDR, which we implemented in a new planner PDRplan[2], performs overall better than the general version of the algorithm combined with encodings (simple sequential and parallel encodings, as well as the $\exists$-step parallel encoding (Rintanen, Heljanko, and Niemelä 2006) and the SASE encoding (Huang, Chen, and Zhang 2012) were tried).
- The *forward* direction of search (progression), where the explicit path is constructed from the initial state towards the goal, gives generally better results than the backward direction (regression). We note that the backward direction is the one preferred in model checking, because it yields superior results for "proving" valid properties. However, when the main focus is on "discovering counterexamples" or, equivalently, on finding plans, the forward direction should be favored.
- Neither *clause propagation* nor *inductive minimization* (Bradley 2011), two features which are normally deemed essential for the performance of PDR, seem to provide a significant benefit on planning problems. This could again be explained by the fact that more than 99 percent of our planning benchmarks are satisfiable (or "solvable").
- When compared to state of the art planners (see Table 1), PDRplan does not reach the performance of Fast Downward (Helmert 2006),[3] but outperforms Mp (Rintanen 2012), probably the current best representative of the planning as satisfiability approach. PDRplan was also able to solve most of the problems on several domains.

[1]http://ipc.icaps-conference.org

[2]https://github.com/quickbeam123/PDRplan

Total	Fast Downward	PDRplan	Mp
1561	1437	1333	1310

Table 1: Number of problems solved within 1800s.

We refer the reader to the full paper (Suda 2014) for an account on plan quality and optimal planning. We also discuss there a perhaps surprising connection between PDR and the Graphplan algorithm of Blum and Furst (1997).

References

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artif. Intell.* 90(1–2):281–300.

Bradley, A. R. 2011. SAT-based model checking without unrolling. In Jhala, R., and Schmidt, D. A., eds., *VMCAI*, volume 6538 of *LNCS*, 70–87. Springer.

Eén, N.; Mishchenko, A.; and Brayton, R. K. 2011. Efficient implementation of property directed reachability. In Bjesse, P., and Slobodová, A., eds., *FMCAD*, 125–134. FMCAD Inc.

Helmert, M. 2006. The Fast Downward planning system. *J. Artif. Intell. Res. (JAIR)* 26:191–246.

Huang, R.; Chen, Y.; and Zhang, W. 2012. SAS$^+$ planning as satisfiability. *J. Artif. Intell. Res. (JAIR)* 43:293–328.

Kautz, H. A., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic and stochastic search. In Clancey, W. J., and Weld, D. S., eds., *AAAI/IAAI, Vol. 2*, 1194–1201. AAAI Press / The MIT Press.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res. (JAIR)* 39:127–177.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artif. Intell.* 170(12–13):1031–1080.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artif. Intell.* 193:45–86.

Suda, M. 2014. Property directed reachability for automated planning. *J. Artif. Intell. Res. (JAIR)*. To appear.

[3]We used the configuration LAMA-2011 (Richter and Westphal 2010), the winner of the satisficing track of IPC 2011.