Exploiting Adjoints in Property Directed Reachability Analysis

Mayuko Kori, Flavio Ascari, Filippo Bonchi, Roberto Bruni, Roberta Gori, and Ichiro Hasuo

We formulate, in lattice-theoretic terms, two novel algorithms inspired by Bradley's property directed reachability algorithm. For finding safe invariants or counterexamples, the first algorithm exploits over-approximations of both forward and backward transition relations, expressed abstractly by the notion of adjoints. In the absence of adjoints, one can use the second algorithm, which exploits lower sets and their principals. As a notable example of application, we consider quantitative reachability problems for Markov Decision Processes.

1 Introduction

Property directed reachability analysis (PDR) refers to a class of verification algorithms for solving safety problems of transition systems [3] [6]. Its essence consists of 1) interleaving the construction of an *inductive invariant* (a *positive chain*) with that of a *counterexample* (a *negative sequence*), and 2) making the two sequences *interact*, with one narrowing down the search space for the other.

PDR algorithms have shown impressive performance both in hardware and software verification, leading to active research [15] [16] [8] [9] going far beyond its original scope. For instance, an abstract domain [4] capturing the over-approximation exploited by PDR has been recently introduced in [7], while PrIC3 [2] extended PDR for quantitative verification of probabilistic systems.

To uncover the abstract principles behind PDR and its extensions, Kori et al. proposed LT-PDR [11], a generalisation of PDR in terms of lattice/category theory. LT-PDR can be instantiated using domain-specific *heuristics* to create effective algorithms for different kinds of systems such as Kripke structures, Markov Decision Processes (MDPs), and Markov reward models. However, the theory in [11] does not offer guidance on devising concrete heuristics.

1.1 Adjoints in PDR.

Our approach shares the same vision of LT-PDR, but we identify different principles: *adjunctions* are the core of our toolset.

An adjunction $f \dashv g$ is one of the central concepts in category theory [13]. It is prevalent in various fields of computer science, too, such as abstract interpretation [4] and functional programming [12]. Our use of adjoints in this work comes in the following two flavours.

$$\begin{tikzcd} A \arrow[r, bend right, "g"'] \arrow[r, phantom, "\perp"] & C \arrow[l, bend right, "f"'] \end{tikzcd}$$

- (forward-backward adjoint) f describes the *forward semantics* of a transition system, while g is the *backward* one, where we typically have $A = C$.

PDR 解析における随伴の活用

Kori, Hasuo, 国立情報学研究所, National Institute of Informatics, Japan.
Kori, Hasuo, 総合研究大学院大学, The Graduate University for Advanced Studies (SOKENDAI), Japan.
Ascari, Bonchi, Bruni, Gori, ピサ大学, Università di Pisa, Italy.

- (abstraction-concretization adjoint) C is a concrete semantic domain, and A is an abstract one, much like in abstract interpretation. An adjoint enables us to convert a fixed-point problem in C to that in A.

1.2 Our Algorithms.

The problem we address is the standard lattice theoretical formulation of safety problems, namely whether the least fixed point of a continuous map b over a complete lattice $(L, \sqsubseteq)$ is below a given element $p \in L$. In symbols $\mu b \sqsubseteq_? p$. We present two algorithms.

The first one, named `AdjointPDR`, assumes to have an element $i \in L$ and two adjoints $f \dashv g \colon L \to L$, representing respectively initial states, forward semantics and backward semantics (see right) such that $b(x) = f(x) \sqcup i$ for all $x \in L$. Under this assumption, we have the following equivalences (they follow from the Knaster-Tarski theorem):

$$\begin{tikzcd} L \arrow[r, bend left, "f"] \arrow[r, phantom, "\perp"] & L \arrow[l, bend left, "g"] \end{tikzcd}$$

$$\mu b \sqsubseteq p \quad \Leftrightarrow \quad \mu(f \sqcup i) \sqsubseteq p \quad \Leftrightarrow \quad i \sqsubseteq \nu(g \sqcap p),$$

where $\mu(f \sqcup i)$ and $\nu(g \sqcap p)$ are, by the Kleene theorem, the limits of the *initial* and *final* chains illustrated below.

$$\perp \sqsubseteq i \sqsubseteq f(i) \sqcup i \sqsubseteq \cdots \qquad \cdots \sqsubseteq g(p) \sqcap p \sqsubseteq p \sqsubseteq \top$$

As positive chain, PDR exploits an over-approximation of the initial chain: it is made greater to accelerate convergence; still it has to be below p.

The distinguishing feature of `AdjointPDR` is to take as a negative sequence (that is a sequential construction of potential counterexamples) an over-approximation of the final chain. This crucially differs from the negative sequence of LT-PDR, namely an under-approximation of the computed positive chain.

We prove that `AdjointPDR` is sound and does not loop but since, the problem $\mu b \sqsubseteq_? p$ is not always decidable, we cannot prove termination. Nevertheless, `AdjointPDR` allows for a formal theory of heuristics that are essential when instantiating the algorithm to concrete problems. The theory prescribes the choices to obtain the boundary executions, using initial and final chains; it thus identifies a class of heuristics guaranteeing termination when answers are negative.

`AdjointPDR`'s assumption of a forward-backward adjoint $f \dashv g$, however, does not hold very often, especially in probabilistic settings. Our second algorithm `AdjointPDR`$^\downarrow$ circumvents this problem by extending the lattice for the negative sequence, from L to the lattice $L^\downarrow$ of *lower sets* in L.

Specifically, by using the second form of adjoints, namely an abstraction-concretization pair, the problem $\mu b \sqsubseteq_? p$ in L can be translated to an equivalent problem on $b^\downarrow$ in $L^\downarrow$, for which an adjoint $b^\downarrow \dashv b^\downarrow_r$ is guaranteed. This allows one to run `AdjointPDR` in the lattice $L^\downarrow$. We then notice that the search for a positive chain can be conveniently restricted to principals in $L^\downarrow$, which have representatives in L. The resulting algorithm, using L for positive chains and $L^\downarrow$ for negative sequences, is `AdjointPDR`$^\downarrow$.

$$\begin{tikzcd} L \arrow[loop left, "b"] \arrow[r, bend right, "(-)^\downarrow"'] \arrow[r, phantom, "\perp"] & L^\downarrow \arrow[l, bend right, "\bigsqcup"'] \arrow[loop right, "b^\downarrow \dashv b^\downarrow_r"] \end{tikzcd}$$

The use of lower sets for the negative sequence is a key advantage. It not only avoids the restrictive assumption on forward-backward adjoints $f \dashv g$, but also enables a more thorough search for counterexamples. `AdjointPDR`$^\downarrow$ can simulate step-by-step LT-PDR, while the reverse is not possible due to a single negative sequence in `AdjointPDR`$^\downarrow$ potentially representing multiple or even all negative sequences in LT-PDR.

1.3 Concrete Instances.

Our lattice-theoretic algorithms yield many concrete instances: the original IC3/PDR [3] [6] as well as Reverse PDR [14] are instances of `AdjointPDR` with L being the powerset of the state space; since LT-PDR can be simulated by $\texttt{AdjointPDR}^{\downarrow}$, the latter generalizes all instances in [11].

As a notable instance, we apply $\texttt{AdjointPDR}^{\downarrow}$ to MDPs, specifically to decide if the maximum reachability probability [1] is below a given threshold. Here the lattice $L = [0,1]^S$ is that of fuzzy predicates over the state space S. Our theory provides guidance to devise two heuristics, for which we prove negative termination. We present its implementation in Haskell, and its experimental evaluation, where comparison is made against existing probabilistic PDR algorithms (PrIC3 [2], LT-PDR [11]) and a non-PDR one (Storm [5]). The performance of $\texttt{AdjointPDR}^{\downarrow}$ is encouraging—it supports the potential of PDR algorithms in probabilistic model checking. The experiments also indicate the importance of having a variety of heuristics, and thus the value of our adjoint framework that helps coming up with those.

Additionally, we found that abstraction features of Haskell allows us to code lattice-theoretic algorithms almost literally ($\sim$100 lines). Implementing a few heuristics takes another $\sim$240 lines. This way, we found that mathematical abstraction can directly help easing implementation effort.

See [10] for further details.

参 考 文 献

[1] Baier, C. and Katoen, J.: *Principles of model checking*, MIT Press, 2008.

[2] Batz, K., Junges, S., Kaminski, B. L., Katoen, J., Matheja, C., and Schröer, P.: PrIC3: Property Directed Reachability for MDPs, *Proc. of CAV 2020, Part II*, Lahiri, S. K. and Wang, C.(eds.), Lecture Notes in Computer Science, Vol. 12225, Springer, 2020, pp. 512–538.

[3] Bradley, A. R.: SAT-Based Model Checking without Unrolling, *Proc. of VMCAI 2011*, Jhala, R. and Schmidt, D. A.(eds.), Lecture Notes in Computer Science, Vol. 6538, Springer, 2011, pp. 70–87.

[4] Cousot, P.: *Principles of Abstract Interpretation*, MIT Press, 2021.

[5] Dehnert, C., Junges, S., Katoen, J., and Volk, M.: A Storm is Coming: A Modern Probabilistic Model Checker, *Proc. of CAV 2017, Part II*, Majumdar, R. and Kuncak, V.(eds.), Lecture Notes in Computer Science, Vol. 10427, Springer, 2017, pp. 592–600.

[6] Eén, N., Mishchenko, A., and Brayton, R. K.: Efficient implementation of property directed reachability, *Proc. of FMCAD 2011*, Bjesse, P. and Slobodová, A.(eds.), FMCAD Inc., 2011, pp. 125–134.

[7] Feldman, Y. M. Y., Sagiv, M., Shoham, S., and Wilcox, J. R.: Property-directed reachability as abstract interpretation in the monotone theory, *Proc. ACM Program. Lang.*, Vol. 6, No. POPL(2022), pp. 1–31.

[8] Gurfinkel, A.: IC3, PDR, and Friends, 2015.

[9] Hoder, K. and Bjørner, N.: Generalized Property Directed Reachability, *Proc. of SAT 2012*, Cimatti, A. and Sebastiani, R.(eds.), Lecture Notes in Computer Science, Vol. 7317, Springer, 2012, pp. 157–171.

[10] Kori, M., Ascari, F., Bonchi, F., Bruni, R., Gori, R., and Hasuo, I.: Exploiting Adjoints in Property Directed Reachability Analysis, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II*, Enea, C. and Lal, A.(eds.), Lecture Notes in Computer Science, Vol. 13965, Springer, 2023, pp. 41–63.

[11] Kori, M., Urabe, N., Katsumata, S., Suenaga, K., and Hasuo, I.: The Lattice-Theoretic Essence of Property Directed Reachability Analysis, *Proc. of CAV 2022, Part I*, Shoham, S. and Vizel, Y.(eds.), Lecture Notes in Computer Science, Vol. 13371, Springer, 2022, pp. 235–256.

[12] Levy, P. B.: *Call-By-Push-Value: A Functional/Imperative Synthesis*, Semantics Structures in Computation, Vol. 2, Springer, 2004.

[13] MacLane, S.: *Categories for the Working Mathematician*, Springer-Verlag, New York, 1971. Graduate Texts in Mathematics, Vol. 5.

[14] Seufert, T. and Scholl, C.: Sequential Verification Using Reverse PDR, *Proc. of MBMV 2017*, Große, D. and Drechsler, R.(eds.), Shaker Verlag, 2017, pp. 79–90.

[15] Seufert, T. and Scholl, C.: Combining PDR and reverse PDR for hardware model checking, *Proc. of DATE 2018*, Madsen, J. and Coskun, A. K.(eds.), IEEE, 2018, pp. 49–54.

[16] Seufert, T. and Scholl, C.: fbPDR: In-depth combination of forward and backward analysis in Property Directed Reachability, *Proc. of DATE 2019*, Teich, J. and Fummi, F.(eds.), IEEE, 2019, pp. 456–461.