Property-Directed Reachability as Abstract Interpretation in the Monotone Theory

YOTAM M. Y. FELDMAN, Tel Aviv University, Israel
MOOLY SAGIV, Tel Aviv University, Israel
SHARON SHOHAM, Tel Aviv University, Israel
JAMES R. WILCOX, Certora, USA

Inferring inductive invariants is one of the main challenges of formal verification. The theory of abstract interpretation provides a rich framework to devise invariant inference algorithms. One of the latest breakthroughs in invariant inference is property-directed reachability (PDR), but the research community views PDR and abstract interpretation as mostly unrelated techniques.

This paper shows that, surprisingly, propositional PDR can be formulated as an abstract interpretation algorithm in a logical domain. More precisely, we define a version of PDR, called Λ-PDR, in which *all* generalizations of counterexamples are used to strengthen a frame. In this way, there is no need to refine frames after their creation, because all the possible supporting facts are included in advance. We analyze this algorithm using notions from Bshouty's monotone theory, originally developed in the context of exact learning. We show that there is an inherent overapproximation between the algorithm's frames that is related to the monotone theory. We then define a new abstract domain in which the best abstract transformer performs this overapproximation, and show that it captures the invariant inference process, i.e., Λ-PDR corresponds to Kleene iterations with the best transformer in this abstract domain. We provide some sufficient conditions for when this process converges in a small number of iterations, with sometimes an exponential gap from the number of iterations required for naive exact forward reachability. These results provide a firm theoretical foundation for the benefits of how PDR tackles forward reachability.

CCS Concepts: • **Theory of computation → Theory and algorithms for application domains**; **Program verification**; • **Software and its engineering → Formal methods**.

Additional Key Words and Phrases: invariant inference, property-directed reachability, abstract interpretation, monotone theory, reachability diameter

ACM Reference Format:
Yotam M. Y. Feldman, Mooly Sagiv, Sharon Shoham, and James R. Wilcox. 2022. Property-Directed Reachability as Abstract Interpretation in the Monotone Theory. *Proc. ACM Program. Lang.* 6, POPL, Article 15 (January 2022), 31 pages. https://doi.org/10.1145/3498676

1 INTRODUCTION

The formal methods community has studied many approaches for automatic verification that are diverse and even seemingly disparate. Two of the main approaches for verifying safety by the automatic inference of inductive invariants are abstract interpretation [Cousot and Cousot 1977] and model checking [Clarke and Emerson 1981; Queille and Sifakis 1982]. State-of-the-art model checking algorithms are based on SAT solving [e.g. Albarghouthi et al. 2012; Graf and Saïdi 1997;

Authors' addresses: Yotam M. Y. Feldman, Tel Aviv University, Israel, yotam.feldman@gmail.com; Mooly Sagiv, Tel Aviv University, Israel, msagiv@acm.org; Sharon Shoham, Tel Aviv University, Israel, sharon.shoham@gmail.com; James R. Wilcox, Certora, USA, james@certora.com.

2475-1421/2022/1-ART15
https://doi.org/10.1145/3498676

Gurfinkel and Ivrii 2017; Gurfinkel et al. 2016; McMillan 2003; Sheeran et al. 2000], with many of the best tools implementing variants of the famous IC3/PDR algorithm [Bradley 2011; Eén et al. 2011], which combines several heuristic ideas to achieve good performance in practice. While PDR is practical and widely deployed, very little is known about it theoretically. In particular, a previous investigation of PDR using the theory of abstract interpretation [Rinetzky and Shoham 2016] had to employ abstractions that are both far from the usual practice of abstract interpreters, and are also too rich in that they can accommodate many algorithms that are unrelated to PDR (see §10). As a result, there is currently no conceptual framework that explains how and when PDR is able to overapproximate and avoid the enumeration of reachable states, which is a key challenge to every invariant inference algorithm.

In this paper, we set out to investigate the principles of how PDR achieves overapproximation. To this end, we continue a line of work that applies learning theory to invariant inference [e.g. Ezudheen et al. 2018; Feldman et al. 2020, 2021; Garg et al. 2014, 2016; Jha et al. 2010; Jha and Seshia 2017; Koenig et al. 2020; Neider et al. 2020; Sharma and Aiken 2016; Sharma et al. 2013b,a, 2012] with a surprising result: the monotone theory from exact learning [Bshouty 1995] enables viewing PDR as classical abstract interpretation (in a new domain). This draws a deep connection between these techniques, and identifies a form of abstraction performed by PDR that distinguishes it both from explicit enumeration and from other algorithmic approaches.

We focus on the fundamental setting of propositional systems, which also applies to infinite-state systems using predicate abstraction [Flanagan and Qadeer 2002; Graf and Saïdi 1997]. PDR constructs a sequence of formulas, called *frames*, by blocking counterexamples (states that can reach bad states). Given a counterexample, the algorithm conjoins to the frame a generalization clause that blocks the counterexample and also additional states, but not states reachable in one step from the previous frame (we explain PDR in detail in §2.2). Theoretically analyzing the behavior of the algorithm is complicated by its highly nondeterministic nature—it depends on the choices of counterexamples and generalization clauses (many of them affected by idiosyncrasies of the underlying SAT solver), and different choices may lead PDR down radically different paths. To ameliorate this, we present an algorithm, called Λ-*PDR*, which resolves this nondeterminism by using all possible answers to these queries, blocking all counterexamples with all admissible generalizations. The resulting frames are tighter than those of PDR, as they include all lemmas that PDR could learn in any execution. This provides a theoretical handhold to study PDR.

Λ-PDR uncovers a key aspect of the generalization performed by standard PDR. The frames are usually viewed as a sequence of overapproximations that prove bounded safety with an increasing bound. While correct, this does not capture the full essence of generalization in PDR. In particular, naive exact forward reachability also computes such a sequence, albeit a trivial one. We show that in Λ-PDR—and hence, in PDR—there is an inherent *abstraction* that includes additional states in each frame beyond exact forward reachability. Applied successively, we show that this is a form of abstract interpretation, which can lead to an exponential gap between the number of frames in Λ-PDR and the number of steps required for exact forward reachability. We prove several results:

(1) We show that the relation between successive frames in Λ-PDR is characterized by an operation from Bshouty's monotone theory [Bshouty 1995]. The idea is that taking all the generalizations that block a state b amounts to computing the least b-monotone overapproximation of the post-image of the previous frame (§4).
(2) We introduce a new abstract domain, of the formulas for which backward reachable states form a monotone basis. We show that Λ-PDR can be viewed as computing Kleene iterations with the best abstract transformer in this domain. Standard PDR also operates in the same domain, and its frames overapproximate the Kleene iterations that Λ-PDR performs. This

```
init  x̄ = (x_n, x_{n−1}, ..., x_0) = 0...0,
      ȳ = (y_n, y_{n−1}, ..., y_0) = 0...0,
      z = 0
repeat: increase_x() | increase_y()
assert ¬(x̄ = 10...0 ∧ ȳ = 11...1 ∧ z = 1)

increase_x():
  if z = 0:
    x̄ = x̄ + 1 (mod 2^{n+1})
    if x̄ = 10...0:
      x̄ = x̄ + 1 (mod 2^{n+1})

increase_y():
  if z = 1:
    ȳ = ȳ + 1 (mod 2^{n+1})
```

Fig. 1. Skip-counter: running example of propositional transition system over the variables $\overline{x} = x_n, \ldots, x_0$. Either `increase_x()` or `increase_y()` is executed in each step according to whether $z = 0$ or $z = 1$, incrementing $\overline{x}$ or $\overline{y}$ resp., but skipping over the value $\overline{x} = 10 \ldots 0$.

is the first time that the theory of state abstraction is able to explain property-directed generalization (§5).

(3) We show exponential gaps between the number of frames in Λ-PDR and the number of iterations of exact forward reachability, as well as the unrolling depth in a dual interpolation-based algorithm (§8).

(4) We prove an upper bound on the number of frames in Λ-PDR in terms of the DNF size of certain "monotonizations" of the transition relation. Although not always tight, this result sheds light on the benefit of the abstraction in certain cases. The proof brings together results from the monotone theory, abstract interpretation, and diameter bounds for transitions systems. This is done by constructing a (hyper)transition system where the states reachable in i steps correspond to the ith Kleene iteration, and bounding the system's diameter (§6–§7).

(5) We show that in some cases the abstraction of Λ-PDR is overly precise, whereas the looser frames of standard PDR converge in fewer and smaller frames (§9).

2 OVERVIEW

2.1 PDR, the Frames

How does property-directed reachability find inductive invariants? Given a set of initial states *Init*, a transition relation δ describing one step of the system, and a set of bad states *Bad*, the goal is to find an *inductive invariant*: a formula I such that $\mathit{Init} \implies I$, $I \implies \neg \mathit{Bad}$, and $\delta(I) \implies I$, where the post-image $\delta(X)$ is the set of states that δ reaches in one step from X.[1] Such an I proves *safety*, that no execution of the system can reach a bad state.

The central data structure that PDR uses to find inductive invariants is the *frames*. These are a sequence of formulas $\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_N$ that satisfies the following properties, for all $0 \le i \le N - 1$:

(1) $\mathcal{F}_0 = \mathit{Init}$; (2) $\mathcal{F}_i \implies \mathcal{F}_{i+1}$; (3) $\delta(\mathcal{F}_i) \implies \mathcal{F}_{i+1}$; (4) $\mathcal{F}_i \implies \neg \mathit{Bad}$.

In words, the frames start with the set of initial states, grow monotonically, always include the states reachable in one step of δ from the previous frame, and are strong enough to prove safety (except possibly the last frame which is "under construction").

These properties ensure that each $\mathcal{F}_i$ overapproximates the set of states reachable in at most i steps, and yet excludes the bad states; this constitutes a proof of *bounded safety*. However, the ultimate goal of PDR is *un*bounded safety, and it is not clear why frames would avoid "overfitting" to bounded executions, and rather converge to a true inductive invariant. In informal discussions, this is sometimes phrased as the criticism that the algorithm merely "happens to find" a bounded safety proof that generalizes to the unbounded case. Indeed, properties 1–4 of frames do not reflect any bias away from bounded proofs, as they are also satisfied by the exact forward reachability algorithm, $\mathcal{F}_0 = \mathit{Init}, \mathcal{F}_{i+1} = \underline{\delta}(\mathcal{F}_i)$, where $\underline{\delta}(X) = X \cup \delta(X)$ denotes the reflexive closure of the post-image. Exact forward reachability might require many frames to converge to an unbounded proof if some states are reachable only by very long paths.

[1] We use a formula and the set of states that satisfy it interchangeably. Unless otherwise stated, the formula to represent a given set of states is chosen arbitrarily.

Consider, for example, the simple family of propositional systems in Fig. 1, parametrized by n. A bit z chooses between incrementing a counter $\overline{x}$ or a counter $\overline{y}$, represented in binary by $x_n, x_{n-1}, \ldots, x_0$ and $y_n, y_{n-1}, \ldots, y_0$ respectively. The safety property to prove is that it is impossible for $\overline{x}$ to have the value $10\ldots0$ while $\overline{y}$ is $11\ldots1$. This property is not inductive as is (for instance, the state $\overline{x} = 10\ldots0, \overline{y} = 11\ldots10, z = 1$ satisfies the safety property but reaches a bad state in one step); one inductive invariant for this system is

$$\overline{x} \neq 10\ldots0 \wedge \overline{y} = 00\ldots0 \wedge z = 0, \tag{1}$$

which implies safety and is closed under a step of the system.

In these systems, exact forward reachability requires $\Omega(2^n)$ iterations before it converges to an inductive invariant. This is because some states, such as $\overline{x} = 10\ldots01, \overline{y} = 00\ldots00, z = 0$, require an exponential number of steps to reach—the system has an exponential *reachability diameter*—so exact forward reachability discovers all reachable states and converges only after that many iterations.

Clearly, invariant inference algorithms must perform some sort of *overapproximation*, or *abstraction*, to overcome this slow convergence. This raises two important questions:

(1) What characterizes the abstraction that PDR performs?
(2) How does this abstraction achieve faster convergence than exact forward reachability?

The commonly-stated properties of frames do not provide an answer; to address these questions we must dive more deeply into how PDR works.

2.2 PDR, the Algorithm

Algorithm 1 PDR

```
1:  procedure PDR(Init, δ, Bad)
2:      F_0^pdr ← Init
3:      N ← 0
4:      while ∀1 ≤ i ≤ N. F_i^pdr ⇏ F_{i-1}^pdr do
5:          F_{N+1}^pdr ← true
6:          while F_{N+1}^pdr ⇏ ¬Bad do
7:              for σ_b ∈ F_{N+1}^pdr ∧ Bad do
8:                  BLOCK(σ_b, N + 1)
9:          N ← N + 1
10:     return F_i^pdr such that F_i^pdr ⟹ F_{i-1}^pdr

11: procedure BLOCK(σ_b, i)
12:     if i = 0 then
13:         unsafe
14:     while δ(F_{i-1}^pdr) ⇏ ¬σ_b do
15:         take σ s.t. σ ⊨ F_i^pdr, (σ, σ_b) ⊨ δ
16:         BLOCK(σ, i − 1)
17:     take c minimal s.t. c ⊆ ¬σ_b and δ(F_{i-1}^pdr) ⟹ c
18:                                  and Init ⟹ c
19:     for 1 ≤ j ≤ i do
20:         F_j^pdr ← F_j^pdr ∧ c
```

Alg. 1 presents a simple version of the basic PDR algorithm. The sequence of frames it manipulates are denoted $\mathcal{F}_0^{\text{pdr}}, \ldots, \mathcal{F}_N^{\text{pdr}}$, to distinguish between PDR's frames and frames of other algorithms in the paper. Initially, $\mathcal{F}_0^{\text{pdr}}$ is initialized to the set of initial states (thereby satisfying property 1). The outer loop terminates once one of the frames is inductive (line 4), which is when $\mathcal{F}_i^{\text{pdr}} \implies \mathcal{F}_{i-1}^{\text{pdr}}$ (because then, from the other properties of frames, $\delta(\mathcal{F}_{i-1}^{\text{pdr}}) \implies \mathcal{F}_i^{\text{pdr}} \implies \mathcal{F}_{i-1}^{\text{pdr}}$). Otherwise, it initializes a new frontier frame to *true* (line 5), and samples bad states (line 7) to block (exclude from the frame) until the frontier frame is strong enough to exclude all bad states (satisfying property 4).

In order to satisfy property 3, before a state σ_b is blocked, the previous frame must be refined it excludes all the pre-states of σ_b (line 14); this is performed by sampling pre-states and blocking them recursively (line 15). Once all the pre-states are blocked in the previous frame, σ_b can be excluded from the current frame. However, at this point, PDR *generalizes* and blocks a *set* of states; this is done by finding *clause* c—also called a *lemma*—that excludes σ_b and still does not exclude any state that is reachable in one step or less from the previous frame (preserving property 3). This is done (in line 18) by starting from all literals (variables or their negations) that are falsified in σ_b, and choosing a subset whose disjunction (which is a clause) satisfies the desired properties.

PDR chooses a *minimal subset* in order to exclude as many states as possible. (In practice, this involves a linear number of SAT calls.)[2] The clause is conjoined (in line 20) to the frame as well as the preceding ones (thereby satisfying property 2, relying on $Init \implies c$).

The above is an operational description of how the frames are generated to be overapproximations, but does not lay bare the principles of why they are computed in this way, and how to characterize the abstraction that frames perform.

To study this, we introduce Λ-PDR[3], an alteration of PDR that is simpler for analysis. This algorithm is a theoretical device to study the abstraction in PDR's frames: each frame of Λ-PDR is tighter than the corresponding frame of PDR, and thus ***the overapproximation that Λ-PDR's frames perform is also performed in usual PDR***. We characterize the abstraction that Λ-PDR performs, and show how it can converge more rapidly than exact forward reachability, which sheds light on the abstraction in PDR.

2.3 Λ-PDR

2.3.1 The Algorithm. Alg. 2 presents Λ-PDR. Briefly, it constructs frames one after the other, by including all possible lemmas that any execution of PDR might learn; $\mathcal{F}_{i+1}$ is the conjunction of *all clauses* that *block* a state from $\mathcal{B}_k$—the set of states that *can reach a bad state* in at most k steps—yet *retain* the states *reachable in one step* from $\mathcal{F}_i$. The algorithm's essentials are similar to PDR's, with important changes.

Algorithm 2 Λ-PDR

```
1:  procedure Λ-PDR(Init, δ, Bad, k)
2:      F_{-1} ← false
3:      F_0 ← Init
4:      B_k = {σ | δ̲^k(σ) ∩ Bad ≠ ∅}
5:      if Init ∩ B_k ≠ ∅ then unsafe
6:      i ← 0
7:      while F_i ⇏ F_{i-1} do
8:          if δ(F_i) ∩ B_k ≠ ∅ then
9:              restart with larger k
10:         F_{i+1} ← true
11:         for σ_b ∈ B_k do
12:             for c ⊆ ¬σ_b do
13:                 if δ̲(F_i) ⟹ c then
14:                     F_{i+1} ← F_{i+1} ∧ c
15:         i ← i + 1
16:     return F_i
```

First, it is useful for our purpose to decouple two roles that frames serve in PDR. One is as a sequence of approximations to the invariant until the frame where an invariant is found (which is usually somewhere in the middle of the sequence). The other is a way to find counterexamples—which are states that can reach a bad state—without unrolling the transition relation [Bradley 2011]. In Λ-PDR we instead imagine that we are provided, through some arbitrary means (such as unrolling), with $\mathcal{B}_k$, the set of states that can reach a state in *Bad* along some execution of length at most k steps (line 4). This allows us to focus on the other role of frames as approximations that converge to the invariant. The number k is chosen in advance, independently of the number of frames N.[4]

Second, frames are computed without backtracking to refine previous frames; lemmas to support future frames are learned eagerly, in advance. In particular, convergence is always at the last frame.

Third, whereas PDR "samples" k-backward reachable states and blocks each counterexample in $\mathcal{F}_{i+1}$ using a single, arbitrary (minimal) clause that does not exclude states in $\underline{\delta}(\mathcal{F}_i)$, Λ-PDR generates *all* such clauses—for *any* counterexample state from $\mathcal{B}_k$ (line 11) and *any* order of dropping literals (line 12). This "determinization" makes the algorithm easier to analyze.

Overall, the algorithm computes each frame $\mathcal{F}_{i+1}$ iteratively, from the previous $\mathcal{F}_i$, without ever refining previous frames. Each frame is the conjunction of all the clauses that can be obtained as lemmas from blocking any counterexample from $\mathcal{B}_k$ while still overapproximating $\underline{\delta}(\mathcal{F}_i)$. This

[2] Practical implementations also attempt to push existing lemmas to other frames whenever possible; we omit this for simplicity. (We discuss inductive generalization below, in §2.7.1.)

[3] In homage to Bshouty's Λ-algorithm [Bshouty 1995], not the SARS-CoV-2 variant.

[4] At first sight $\mathcal{F}_i^{\text{pdr}}$ consists of clauses that exclude counterexamples from a lower backward reachability bound, $\mathcal{B}_{N-i}$; but in fact, pushing lemmas forward means that even $\mathcal{F}_N^{\text{pdr}}$ can include clauses learned at $\mathcal{F}_1^{\text{pdr}}$ from counterexamples in $\mathcal{B}_N$.

process continues until an inductive invariant is found (line 7)—unless the current frame does include a counterexample from $\mathcal{B}_k$, which prompts an increase of k (line 8) in order to distinguish between spurious overapproximation and truly unsafe systems (detected in line 5 by an initial state that can reach a bad state in k steps).

As an example, this is how Λ-PDR proceeds on the example of Fig. 1 with (say) $k = 1$: The k-backward reachable states $\mathcal{B}_k$ are those where $\overline{x} = 10\ldots00 \wedge y_n y_{n-1}\ldots y_1 = 11\ldots1 \wedge z = 1$ (every value of y_0 yields a backward reachable state). The frame sequence is initialized with $\mathcal{F}_0 = \mathit{Init}$. As $\underline{\delta}(\mathcal{F}_0)$ does not intersect $\mathcal{B}_k$, the algorithm proceeds to computing $\mathcal{F}_1$. It starts as *true*, and the algorithm iterates through the states in $\mathcal{B}_k$ to generate clauses. Suppose that the first counterexample σ_b is $\overline{x} = 10\ldots00 \wedge \overline{y} = 11\ldots10 \wedge z = 1$. We write $\neg\sigma_b = (x_n \neq 1) \vee (x_{n-1} \neq 0) \vee \ldots \vee (x_1 \neq 0) \vee (x_0 \neq 0) \vee (y_n \neq 1) \vee (y_{n-1} \neq 1) \vee \ldots \vee (y_1 \neq 1) \vee (y_0 \neq 0) \vee (z \neq 1)$, and consider every possible sub-clause c thereof, checking whether $\underline{\delta}(\mathcal{F}_0) \implies c$, namely, c includes both $\overline{x} = 00\ldots00, \overline{y} = 00\ldots00, z = 0$ and $\overline{x} = 00\ldots01, \overline{y} = 00\ldots00, z = 0$. In this case, there are several incomparable (and minimal) such c's: $x_n \neq 1, y_i \neq 1$ for every $i > 1$, and $z \neq 1$. In Λ-PDR, *all* these potential clauses are conjoined to $\mathcal{F}_1$. The algorithm performs the same procedure for all the counterexamples in $\mathcal{B}_k$. Once this is done, $\mathcal{F}_1$ never changes again in the course of the algorithm, and it becomes the basis for constructing $\mathcal{F}_2$ in the same way, and so on until an inductive invariant is found or a restart becomes necessary. (We later show the resulting $\mathcal{F}_1, \mathcal{F}_2, \ldots$ in this example.)

2.3.2 PDR Overapproximates at Least as Much as Λ-PDR. The importance of Λ-PDR for our investigation of abstraction in PDR stems from the fact that $\mathcal{F}_i \implies \mathcal{F}_i^{\text{pdr}}$, when $k = N$ is the final number of frames in PDR (Corollary 9.3). This implies that whatever overapproximation Λ-PDR performs also transfers to PDR: the overapproximation in Λ-PDR is a lower bound for the overapproximation in PDR. The relationship $\mathcal{F}_i \implies \mathcal{F}_i^{\text{pdr}}$ holds because every clause c that PDR can use to strengthen $\mathcal{F}_i^{\text{pdr}}$ is used to strengthen $\mathcal{F}_i$ of Λ-PDR (roughly, in PDR, for c to added to $\mathcal{F}_i^{\text{pdr}}$, it must block a counterexample from $\mathcal{B}_k$ and overapproximate the post-image of the previous frame; the same properties would hold for c in Λ-PDR, thus ensuring that c is conjoined to $\mathcal{F}_i$—see Corollary 9.3).

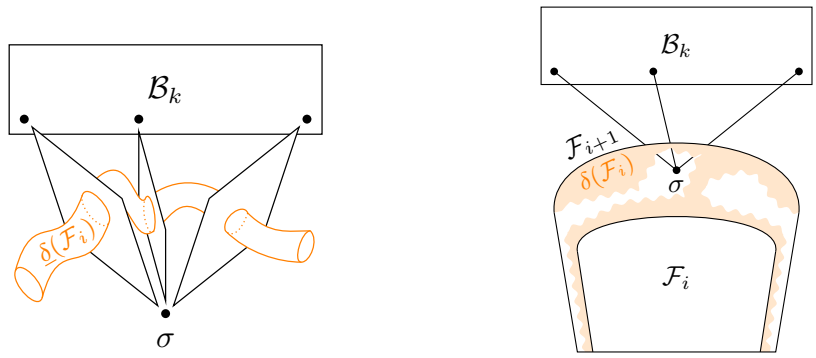
Our goal then is to show that Λ-PDR performs significant overapproximation over exact forward reachability, and thereby establish the same for PDR. Our first step is to characterize the overapproximation that Λ-PDR performs, and for this we need tools developed in exact concept learning.

2.4 Abstraction from The Monotone Theory

The main technical enabler of our paper is the observation (Corollary 4.10) that in Λ-PDR, there is a well-defined relation between successive frames, through what we call the *monotone hull*:

$$\mathcal{F}_{i+1} = \text{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i)) \stackrel{\text{def}}{=} \bigwedge_{b \in \mathcal{B}_k} \mathcal{M}_b(\underline{\delta}(\mathcal{F}_i)), \tag{2}$$

where $\mathcal{M}_b(\varphi)$ is the central operator in the monotone theory [Bshouty 1995], the *least b-monotone overapproximation of φ* ("b-monotonization" in short). A Boolean function f is b-monotone, when b is a state, if whenever $f(\sigma_1) = 1$ and $\sigma_1 \leq_b \sigma_2$, which means that σ_2 is obtained from σ_1 by flipping bits on which σ_1, b agree, then also $f(\sigma_2) = 1$. $\mathcal{M}_b(\varphi)$ is the *least* b-monotone formula (function) implied by φ. (We elaborate on the technical details in §4.1.) The insight of Equation (2) is that, as we show, every lemma in PDR is implied by the monotone hull, and the conjunction of all possible lemmas is exactly the monotone hull. Technically, the observation builds on an equivalent formulation of $\mathcal{M}_b(\varphi)$ in a conjunctive form, which is not explicit in Bshouty's paper (Thm. 4.9).



(a) (b)

Fig. 2. (a) σ is "protected" by $\underline{\delta}(\mathcal{F}_i)$ from exclusion due to blocking $\mathcal{B}_k$, thus (b) σ is in $\mathcal{F}_{i+1} = \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i))$.

Our central observation is that the monotone hull operator introduces *overapproximation* to the sequence of frames—$\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i))$ can include many more states than $\underline{\delta}(\mathcal{F}_i)$. This is an interesting deviation from Bshouty's use of monotonizations in *exact* learning of an unknown φ, where a set B is chosen such that $\varphi \equiv \mathrm{MHull}_B(\varphi)$; in that case B is said to be a monotone basis for φ, denoted $\varphi \in \mathrm{MSpan}(B)$ (Def. 5.2). In contrast, here the monotone hull is applied to intermediate frames, and we are interested in the cases that the set $\mathcal{B}_k$ is *not* a monotone basis for $\underline{\delta}(\mathcal{F}_i)$, for then Equation (2) indicates a strict *overapproximation of exact forward reachability* that Λ-PDR performs.

Consider again the running example (Fig. 1). One step of exact forward reachability discovers the state $\overline{x} = 00\ldots01, \overline{y} = 00\ldots00, z = 0$ in addition to the initial state. In contrast, by Equation (2), the first frame of Λ-PDR with $k = 1$ is $\mathcal{F}_1 = \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathit{Init}))$, resulting in

$$\mathcal{F}_1 \;=\; x_n = 0 \wedge \overline{y} = 00\ldots00 \wedge z = 0; \tag{3}$$

in a single iteration the algorithm has leaped over an exponential number of steps of δ!

To compute $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathit{Init}))$, in order to arrive at Equation (3), we compute the monotone overapproximations. In our example, $\mathcal{B}_k$ is a single cube:

$$\mathcal{B}_k = (x_n = 1 \wedge x_{n-1} = 0 \wedge \ldots \wedge x_1 = 0 \wedge x_0 = 0 \wedge y_n = 1 \wedge \ldots \wedge y_1 = 1 \wedge z = 1),$$

in which case $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_0))$ can be calculated by writing $\underline{\delta}(\mathcal{F}_0)$ in DNF (see Lemmas 4.5 and 4.8):

$$\begin{aligned}\underline{\delta}(\mathcal{F}_0) = (&x_n = 0 \wedge x_{n-1} = 0 \wedge \ldots \wedge x_1 = 0 \wedge x_0 = 0 \wedge y_n = 0 \wedge \ldots \wedge y_1 = 0 \wedge y_0 = 0 \wedge z = 0)\\ \vee\, (&x_n = 0 \wedge x_{n-1} = 0 \wedge \ldots \wedge x_1 = 0 \wedge x_0 = 1 \wedge y_n = 0 \wedge \ldots \wedge y_1 = 0 \wedge y_0 = 0 \wedge z = 0),\end{aligned}$$

and in each term omitting every literal that agrees with the cube of $\mathcal{B}_k$ (appearing in color). (When $\mathcal{B}_k$ is not a single cube, $\mathcal{F}_{i+1}$ is computed as a conjunction of the above for each cube in $\mathcal{B}_k$.)

What is especially significant about this overapproximation in Λ-PDR is that it exists *in each step*, as we describe in the next subsection (§2.5). But before that, we explain the intuition for where this overapproximation stems from.

The cause of overapproximation in Λ-PDR is the special constraints on the lemmas the algorithm can generate. Recall that the states that remain in $\mathcal{F}_{i+1}$ are those that *cannot be excluded* by *any* lemma starting from *any* counterexample, due to the need to satisfy property 3. Since lemmas are *not* arbitrary formulas, perhaps surprisingly, such states exist beyond the exact post-image. We demonstrate what these states are using the running example. Consider a state σ that satisfies Equation (3). Why does no lemma c learned by the algorithm exclude σ, i.e., $\sigma \not\models c$? The reason is that every c excludes some counterexample $b \in \mathcal{B}_k$, and, furthermore, c is a clause—a

negation of a cube. A cube is a very rigid geometric shape; if $\neg c$ contains both σ and b then it necessarily contains many other states—it must include all states that are within the smallest cube that contains both σ, b, a.k.a. the Hamming interval between σ, b [e.g. Wiedemann 1987]. For example, the Hamming interval between $\sigma = (\overline{x} = 011\ldots101, \overline{y} = 00\ldots00, z = 0)$ and $b = (\overline{x} = 10\ldots00, \overline{y} = 11\ldots10, z = 1)$ is $x_1 = 0 \wedge y_0 = 0$—the conjunction of the literals (or constraints) that hold in both σ, b. However, $\neg c$ must not contain any state in $\underline{\delta}(\mathcal{F}_i)$, so the Hamming interval between σ, b cannot intersect $\underline{\delta}(\mathcal{F}_i)$. In our example, the Hamming interval between σ and b includes the state $\widetilde{\sigma} = (\overline{x} = 00\ldots00, \overline{y} = 00\ldots00, z = 0)$, and $\widetilde{\sigma} \in \underline{\delta}(\mathcal{F}_0)$, so a lemma c that excludes σ and originates from blocking b cannot be conjoined to $\mathcal{F}_1$.

Put differently, $\widetilde{\sigma} \in \underline{\delta}(\mathcal{F}_0)$ "protects" $\sigma \in \mathcal{F}_1$ from being excluded due to b. In general, a state σ is included in $\mathcal{F}_{i+1}$ if a protector state $\widetilde{\sigma} \in \mathcal{F}_i$ exists *for every* $b \in \mathcal{B}_k$, namely, the Hamming interval between σ, b crosses $\underline{\delta}(\mathcal{F}_i)$ for all b's (Fig. 2). In our example, the same $\widetilde{\sigma}$ actually protects every $\sigma \in \mathcal{F}_1$ from exclusion due to any $b \in \mathcal{B}_k$, but multiple protector states may be necessary in general.

The idea of protector states explains why $\mathcal{F}_{i+1}$ is $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i))$ (Equation (2)). Every state $\widetilde{\sigma} \in \underline{\delta}(\mathcal{F}_0)$ is a protector state. The states that $\widetilde{\sigma}$ protects from b are the states σ such that $\widetilde{\sigma}$ is in the Hamming interval between σ and b; these states are "farther away" from b than $\widetilde{\sigma}$, in the sense of $\widetilde{\sigma} \leq_b \sigma$ as defined above (and formally in Def. 4.1). The set protected from b by $\underline{\delta}(\mathcal{F}_i)$ is therefore $\mathcal{M}_b(\underline{\delta}(\mathcal{F}_i))$, and the states that are protected from all $b \in \mathcal{B}_k$ are the conjunction over all b's, namely $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i))$.

2.5 Successive Overapproximation: Abstract Interpretation

The overapproximation of Equation (2) is present between each two consecutive frames; it is thus performed repeatedly, using the previous overapproximation as the starting point of the next. In §5 we show that Λ-PDR can be cast as ***abstract interpretation in a new logical domain***, of the formulas in $\mathrm{MSpan}(\mathcal{B}_k)$, the formulas φ s.t. $\mathrm{MHull}_{\mathcal{B}_k}(\varphi) \equiv \varphi$, which are the formulas expressible by a conjunction of clauses that each excludes a state from $\mathcal{B}_k$ (Def. 5.2). ***The frames of Λ-PDR are completely characterized as Kleene iterations with the best abstract transformer*** in this domain (Thm. 5.6), when correcting for the slightly different initial frame ($\mathcal{F}_0 = \mathit{Init}$ vs. $\mathrm{MHull}_{\mathcal{B}_k}(\mathit{Init})$; we show that the resulting difference in the number of frames is at most one).

Repeatedly applying the abstraction can cause Λ-PDR to converge much faster than exact forward reachability (illustrated schematically in Fig. 3). For example, the frames of Λ-PDR on the running example are displayed in Fig. 4 (we perform the calculation in detail in Example 4.11), and $\mathcal{F}_3$ is none other than the inductive invariant of Equation (1). In this way, ***successive overapproximation can lead to convergence in a smaller number of frames than exact forward reachability***; in this example, Λ-PDR converges in 4 frames, rather than an exponential number as exact forward reachability would use.[5] In §8, we show that Λ-PDR holds a similar advantage over the unrolling depth of an interpolation-based algorithm.

2.6 Convergence Bounds via (Hyper)Diameter Bounds

When does the successive overapproximation of Λ-PDR terminate in a small number of iterations? The lattice height of the abstract domain is exponential in the number of variables (see §5), and rapid convergence depends on properties of the transition system rather than of the abstract domain.

We prove a convergence bound using one such property: ***when the DNF size*** (the number of terms in the smallest DNF representation) ***of specific monotone overapproximations of the transition relation are small, then Λ-PDR terminates after a small number of iterations*** (Thms. 6.2 and 7.2).

[5] The operations in the abstract domain are not efficient; we focus on the number of iterations until convergence.

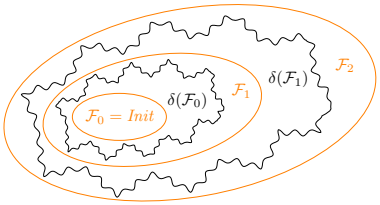


Fig. 3. Repeatedly interleaving the post-image and monotone hull operators results in successive overapproximation.

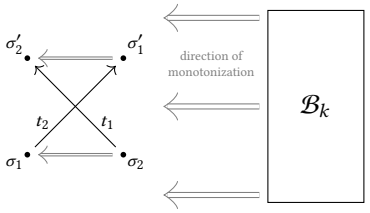


Fig. 5. The monotonization of a transition $t_1 = (\sigma_1, \sigma'_1) \in \delta$ subsumes that of $t_2 = (\sigma_2, \sigma'_2) \in \delta$ if σ_1 is farther from the backward reachable cube than σ_2, and σ'_1 is closer than σ'_2. If few transitions subsume the rest, $\left|\delta^{\divideontimes}\right|_{\text{dnf}}$ is small, which implies convergence with few frames for Λ-PDR.

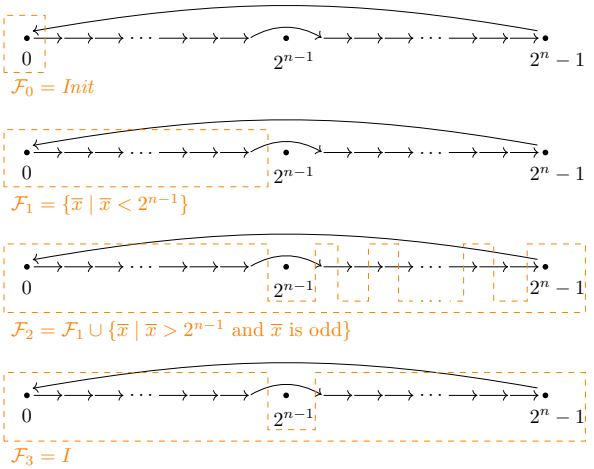


Fig. 4. The frames of Λ-PDR on the running example (only values of $\overline{x}$ are displayed, always $\overline{y} = 0 \ldots 0, z = 0$). The number of frames required for convergence is always 4, independent of the parameter n.

Back in the example, we bound the number of iterations of Λ-PDR by applying Thm. 6.2 to the part of δ restricted to states where $\overline{y} = 0 \ldots 0, z = 0$ (this is valid because both the transitions of δ and monotonization w.r.t. $\mathcal{B}_k$, $\mathcal{M}_{\overline{x}=10\ldots0,\overline{y}=1\ldots1,z=1}(\cdot)$, leave $\overline{y} = 0 \ldots 0, z = 0$ unchanged—see Remark 7.1). The transition relation $\tilde{\delta} = \delta\big|_{\overline{y}\leftarrow 00\ldots00, z\leftarrow 0}$ can be written in DNF (as a double-vocabulary formula, with unprimed variables for the pre-state and primed variables for the post-state) as the disjunction of individual transitions (colors and boxes should be ignored at this point):

$$
\begin{aligned}
\tilde{\delta} = {} & \boxed{(\overline{x} = 000\ldots0000 \wedge \overline{x}' = 000\ldots0001)} \\
\vee {} & \boxed{(\overline{x} = 000\ldots0001 \wedge \overline{x}' = 000\ldots0010)} \vee (\overline{x} = 000\ldots0010 \wedge \overline{x}' = 000\ldots0011) \\
\vee {} & \boxed{(\overline{x} = 000\ldots0011 \wedge \overline{x}' = 000\ldots0100)} \vee (\overline{x} = 000\ldots0100 \wedge \overline{x}' = 000\ldots0101) \\
\vee {} & (\overline{x} = 000\ldots0101 \wedge \overline{x}' = 000\ldots0110) \vee (\overline{x} = 000\ldots0110 \wedge \overline{x}' = 000\ldots0111) \\
\vee {} & \boxed{(\overline{x} = 000\ldots0111 \wedge \overline{x}' = 000\ldots1000)} \vee \ldots \\
\vee {} & \boxed{(\overline{x} = 011\ldots1111 \wedge \overline{x}' = 100\ldots0001)} \vee \ldots \\
\vee {} & (\overline{x} = 111\ldots1110 \wedge \overline{x}' = 111\ldots1111) \\
\vee {} & \boxed{(\overline{x} = 111\ldots1111 \wedge \overline{x}' = 000\ldots0000)}.
\end{aligned}
$$

Obviously, the number of terms in $\tilde{\delta}$ is $\Omega(2^n)$. To analyze Λ-PDR, we perform a monotonization of the (two-vocabulary) transition relation. Recall that in this example $\mathcal{B}_k$ consists of a single

cube, in which case we need only consider one monotonization. (The analysis with more complex syntactic forms of $\mathcal{B}_k$ is sketched below and at length in §7.2.) We examine the monotonization that omits literals that agree with $\mathcal{B}_k$ in the post-state, and *conversely* in the pre-state: $\delta^{\times} = \mathcal{M}_{\overline{x}=01\ldots11,\overline{x}'=10\ldots00}(\tilde{\delta})$. The literals in $\tilde{\delta}$ that are omitted in $\delta^{\times}$ appear colored. The monotonization term-by-term from δ creates many redundant terms; the terms in the boxes above comprise a succinct DNF representation of $\delta^{\times}$. Essentially, when $x_n = 0$, the transitions that create a new leading 1 subsume all later transitions with the same leading 1; similarly for when $x_n = 1$; and there are also two additional "special" transitions to include—skipping the value $10\ldots0$ and wraparound from $11\ldots1$ to $00\ldots0$. Overall, this is merely $O(n)$ terms in $\delta^{\times}$. By Thm. 6.2 we deduce from the linear DNF size of $\delta^{\times}$ that Λ-PDR converges in a linear number of iterations.

One way to think about the difference between $\delta, \delta^{\times}$ is by the way transitions in δ give rise to the transitions in $\delta^{\times}$. Effectively, $\delta^{\times}$ builds on the transitions of δ, adding to them abstraction steps before and after: a transition of $\delta^{\times}$ can move away from $\mathcal{B}_k$, follow a concrete transition of δ, and from the resulting post-state again move in the direction away from $\mathcal{B}_k$. This is illustrated in Fig. 5, where $\delta^{\times}$ can use the transition (σ_1, σ_1') to arrive from σ_2 to σ_2', even if (σ_2, σ_2') were not a transition of δ. How much each abstraction susbstep moves away from $\mathcal{B}_k$ can differ, and so every transition of δ induces a *set* of transitions in $\delta^{\times}$ that use it. The set of transitions of $\delta^{\times}$ induced by a transition $(\sigma_1, \sigma_1') \in \delta$ can subsume the set induced by another transition $(\sigma_2, \sigma_2') \in \delta$, as depicted in Fig. 5. In that case, when performing the monotonization of δ to obtain $\delta^{\times}$, the term generated by (σ_1, σ_2') can be discarded, because it is subsumed by the term generated by (σ_1, σ_1'), which can lead to fewer terms in $\delta^{\times}$. Roughly, Thm. 6.2 shows that Λ-PDR converges rapidly whenever there is a small number of transitions that subsume the others, by going from a pre-state σ that is "very far" from $\mathcal{B}_k$ in Hamming distance compared to the pre-states of other transitions, to the post-state σ' that is "very close" to $\mathcal{B}_k$ compared to the post-states of other transitions. This is an intuition for how a small $\left|\delta^{\times}\right|_{\text{dnf}}$ can arise from the monotonization of the fully-expanded DNF representation of δ. The starting point for monotonization can also be a more succinct DNF representation of δ, in which case the intuition for an even shorter DNF representation of $\delta^{\times}$ is similar.

To prove the theorem, given a transition relation δ, we show that the reachable states of $\delta^{\times}$ match the frames of the Kleene iterations with the best transformer for δ. In particular, the *reachability diameter* [Biere et al. 1999a] of the resulting abstract system matches the number of frames in Λ-PDR with a difference of at most 1. Hence, ***bounds on the diameter of the abstract system result in bounds on the number of iterations***. We then bound the diameter by the DNF size of the transition relation $\delta^{\times}$ above, resulting overall in a bound on the number of iterations of Λ-PDR.

For when $\mathcal{B}_k$ consists of multiple cubes, we generalize Thm. 6.2 to Thm. 7.2, bounding the number of frames by a product of monotonizations of δ and *Init*. In the proof, the construction involves not an ordinary transition system, but a *hyper*transition system: the hypertransition relation $\delta^{\times}$ arrives through concrete transitions to a *set* of states, and abstracts from them to a state "protected" by that set, because the abstraction requires a "protector" state from every state in $\mathcal{B}_k$ (see Fig. 2). A similar diameter bound using the DNF size of the hypertransition relation $\delta^{\times}$ applies. We show that $\delta^{\times}$ can be written as a conjunction of per-cube monotonizations, leading to a bound by the product of DNF sizes of monotonizations of δ and *Init* (see §7).

This technique does not explain rapid convergence of Λ-PDR in full generality, but provides one explanation for how the abstraction can bring this about.

2.7 PDR, Revisited

Through Λ-PDR, we have shown how PDR's frames perform an abstract interpretation in a domain founded on the monotone theory, and how such an abstraction can lead to faster convergence. We

observe that these important characteristics of PDR are concealed in a simple property of PDR's frames: that they can be written in CNF so that every clause excludes at least one state from $\mathcal{B}_N$ (Lemma 9.2). In the monotone theory from above this reads that for every $1 \leq i \leq N$,

(5) $\mathcal{F}_i \in \mathrm{MSpan}(\mathcal{B}_N)$.

The frames of Λ-PDR are the least set of states that satisfy this property together with properties 1–4 from §2.1 (Lemma 9.1), and the frames of PDR overapproximate them (Corollary 9.3). Property 5 is the regularization in our abstract domain (§5), and we have shown that it can lead to faster convergence than exact post-image computations—although PDR does not necessarily converge in the same number of frames as Λ-PDR, due to its additional overapproximation and heuristics.

The fact that PDR's frames are not the least to satisfy the above properties can have some benefits. We show two:

- **Faster convergence**: In some cases Λ-PDR performs little or no abstraction over exact forward reachability, but the fact that PDR only samples a subset of the possible lemmas can guarantee fast convergence. We show an example where Λ-PDR requires an exponential number of frames, whereas a linear number always suffices for standard PDR.
- **Frame size**: Λ-PDR's frames may be (needlessly) complex to represent as a formula. We show an example where some frames of Λ-PDR necessarily have an exponential DNF or CNF size, whereas standard PDR can converge in the same number of frames that include only a small number of important lemmas.

2.7.1 Discussion: Additional PDR Features. Our study focuses on what is, in our view, the most basic version of PDR. Our approach provides an interesting starting point to a discussion of two common, more advanced features of PDR.

Other forms of generalization. Inductive generalization [Bradley 2011] minimizes lemmas using a stronger criterion: a lemma c can be learned in $\mathcal{F}_{i+1}^{\mathrm{pdr}}$ if it is inductive *relative* to $\mathcal{F}_i$—whether $\underline{\delta}(\mathcal{F}_i^{\mathrm{pdr}} \wedge c) \implies c$, namely, checking whether c holds in the post-state while also assuming c in the pre-state. At first sight, this feature is not present in Λ-PDR, which uses the standard check (Alg. 2, line 13). Surprisingly, lemmas that PDR can generate using inductive generalization are also present in Λ-PDR (with $k = N$). This is a consequence of the fact that PDR with inductive generalization still satisfies properties 1–4 [Bradley 2011; Eén et al. 2011], as well as property 5. The optimization of inductive generalization becomes important only when lazily backtracking to refine previous frames. Other techniques, such as ternary simulation [Eén et al. 2011], propagate sets of states to block together. If any of the states is in $\mathcal{B}_k$, the resulting lemma is present also in Λ-PDR.

May-counterexamples. Some variants of PDR produce lemmas by blocking may counterexamples [Gurfinkel and Ivrii 2015] that are not necessarily backward reachable, as a way to encourage pushing existing lemmas to later frames. In Λ-PDR, all admissible lemmas from $\mathrm{MSpan}(\mathcal{B}_k)$ are always included, hence may counterexamples are not useful for pushing such lemmas. However, may counterexamples also mean that lemmas no longer necessarily block states in $\mathcal{B}_N$, which could be beneficial if a large N is required to have an inductive invariant $I \in \mathrm{MSpan}(\mathcal{B}_N)$. (It is a necessary condition PDR; in Λ-PDR it is both necessary and sufficient, see Corollary 5.8.) This could be thought of as (heuristically) increasing the set $\mathcal{B}_k$. In Λ-PDR, this results in a richer abstract domain that includes more inductive invariants but leads to tighter frames with less overapproximation. The theoretical ramifications of this beyond Λ-PDR merit more study.

2.7.2 Outline. The rest of the paper is organized as follows: §3 introduces preliminary notation. §4 introduces notions from the monotone theory and establishes their connection to Λ-PDR. §5 proves the connection to abstract interpretation. §6 develops a bound on the number of iterations of Λ-PDR

for the case that $\mathcal{B}_k = b$ a single cube, and §7 generalizes these results to arbitrary $\mathcal{B}_k$. §8 contrasts forward reachability in Λ-PDR with exact forward reachability and a dual interpolation-based algorithm. §9 compares Λ-PDR to standard PDR. §10 discusses related work, and §11 concludes.

3 PRELIMINARIES

We consider the safety of transition systems defined over propositional vocabularies.

States, transition systems, inductive invariants. Given a vocabulary $\Sigma = \{p_1, \ldots, p_n\}$ of n Boolean variables, a *state* is a *valuation* to Σ. The set of states over Σ is denoted States$[\Sigma]$. If x is a state, $x[p]$ is the value (*true*/*false* or 1/0) that x assigns to the variable $p \in \Sigma$. A *transition system* is a triple $(Init, \delta, Bad)$ where $Init$, Bad are formulas over Σ denoting the set of initial and bad states (respectively), and the *transition relation* δ is a formula over $\Sigma \uplus \Sigma'$, where $\Sigma' = \{x' \mid x \in \Sigma\}$ is a copy of the vocabulary used to describe the post-state of a transition. If $\tilde{\Sigma}, \tilde{\Sigma}'$ are distinct copies of Σ, $\delta[\tilde{\Sigma}, \tilde{\Sigma}']$ denotes the substitution in δ of each $p \in \Sigma$ by its corresponding in $\tilde{\Sigma}$ and likewise for $\Sigma', \tilde{\Sigma}'$. Given a set of states $S \subseteq$ States$[\Sigma]$, the post-image $\delta(S) = \{\sigma' \mid \exists \sigma \in S.\ (\sigma, \sigma') \models \delta\}$. The reflexive post-image is $\underline{\delta}(S) = \delta(S) \cup S$. The reachability diameter of a system is the least s s.t. if σ is reachable from $Init$ by δ in any number of steps, it is reachable in at most s steps. The set of *k-backward reachable states*—those that can reach a state in Bad along some execution of length at most k—are $\mathcal{B}_k = \{\sigma \mid \underline{\delta}^k(\{\sigma\}) \cap Bad \neq \emptyset\}$. A transition system is *safe* if all the states that are reachable from $Init$ via any number of steps of δ satisfy $\neg Bad$. An *inductive invariant* is a formula I over Σ such that (i) $Init \implies I$, (ii) $I \wedge \delta \implies I'$, and (iii) $I \implies \neg Bad$, where I' denotes the result of substituting each $x \in \Sigma$ for $x' \in \Sigma'$ in I, and $\varphi \implies \psi$ denotes the validity of the formula $\varphi \rightarrow \psi$. In the context of propositional logic, a transition system is safe iff it has an inductive invariant.

CNF, DNF, and cubes. A *literal* ℓ is a variable p or its negation $\neg p$. A *clause* c is a disjunction of literals. The empty clause is *false*. A formula is in *conjunctive normal norm (CNF)* if it is a conjunction of clauses. A *cube* or *term* d is a conjunction of a consistent set of literals; at times, we also refer directly to the set and write $\ell \in d$. The empty cube is *true*. A formula is in *disjunctive normal form (DNF)* if it is a disjunction of terms. The *domain*, $dom(d)$, of a cube d is the set of variables that appear in it (positively or negatively). Given a state σ, we use the state and the (full) cube that consists of all the literals that are satisfied in σ interchangeably; the only satisfying valuation of the cube is σ. We identify a formula with the set of its valuations, and a set of valuations with an arbitrary formula that represents it, chosen arbitrarily (which always exists in propositional logic).

4 THE MONOTONE THEORY FOR Λ-PDR

In this section, we present the monotone theory by Bshouty [1995] and our extensions, and use it to derive the relation between successive frames in Λ-PDR (Equation (2)). §4.1 defines *least b-monotone overapproximations* $\mathcal{M}_b(\cdot)$. §4.2 defines the *monotone hull* $\text{MHull}_B(\cdot)$ w.r.t. a set of states B, which is a conjunction of monotone overapproximations, and then relates this to Λ-PDR. All omitted proofs appear in the extended version [Feldman et al. 2022].

4.1 Monotone Overapproximations

Our definitions and claims concerning b-monotone overapproximations generalize Bshouty [1995] by considering a partial cube b, and coincide with the original in the case of a full cube.

Definition 4.1 (b-Monotone Order [Bshouty 1995]). Let b be a cube. We define a partial order over states where $v \leq_b x$ when x, v agree on all variables not present in b, and x disagrees with b on all variables on which also v disagrees with b: $\forall p \in \Sigma.\ x[p] \neq v[p]$ implies $p \in dom(b) \wedge v[p] = b[p]$.

Geometrically, $v \leq_b x$ indicates that x is "farther away" from b in the Hamming cube than v from b, namely, that there is a shortest path w.r.t. Hamming distance between x and its projection onto b that goes through v.

Definition 4.2 (b-Monotonicity [Bshouty 1995]). A formula ψ is b-monotone for a cube b if $\forall v \leq_b x.\ v \models \psi$ implies $x \models \psi$.

That is, if v satisfies ψ, so do all the states that are farther away from b than v. For example, if ψ is 000-monotone and $100 \models \psi$, then because $100 \leq_{000} 111$ (starting in 100 and moving away from 000 can reach 111), also $111 \models \psi$. In contrast, $100 \not\leq_{000} 011$ (the same process cannot flip the 1 bit that already disagrees with 000), so 011 does not necessarily belong to ψ.

The importance of b-monotonicity for us is that if $\sigma \models b$, then, as we shall see (in Thm. 4.9), any clause $c \subseteq \neg\sigma$ is a b-monotone formula. The reason is that if $v \models c$ and we flip variables to disagree more with b, we can only make v satisfy more literals of c than before: all the variables in b appear in the opposite polarity in $\neg\sigma$ and hence also in c, so flipping them in $v \models c$ to disagree with b makes them agree with c even more, and the result also satisfies c.

Further, the lemmas that PDR learns are not just clauses, but clauses that overapproximate a certain set, hence they are b-monotone overapproximations. There are many b-monotone overapproximations, but our main workhorse is the least such formula:

Definition 4.3 (Least b-Monotone Overapproximation [Bshouty 1995]). Given a formula φ and a cube b, the *least b-monotone overapproximation* of φ is a formula $\mathcal{M}_b(\varphi)$ defined by

$$x \models \mathcal{M}_b(\varphi) \text{ iff } \exists v.\ v \leq_b x \wedge v \models \varphi.$$

For example, if $100 \models \varphi$, then $100 \models \mathcal{M}_{000}(\varphi)$ because $\mathcal{M}_{000}(\varphi)$ is an overapproximation, and hence $111 \models \mathcal{M}_{000}(\varphi)$ because it is 000-monotone, as above. Here, thanks to minimality, 011 does not belong to $\mathcal{M}_{000}(\varphi)$, unless 000, 001, 010, or 011 belong to φ.

$\mathcal{M}_b(\varphi)$ is a well-defined overapproximation of φ. Its main significance for learning theory is that it can be computed efficiently (through the DNF representation we also show below), obtaining the original φ as the conjunction of least b-monotone overapproximations (with different b's). Surprisingly, the same overapproximation is related to PDR; we show below that $\mathcal{M}_b(\varphi)$ is exactly the conjunction of all the clauses c that overapproximate φ and can arise from blocking a state in b (Thm. 4.9), matching generalization in the construction of frames of Λ-PDR (Corollary 9.3).

A technical observation that will prove useful several times is that $\mathcal{M}_b(\cdot)$ is a monotone operator:

Lemma 4.4. *If $\varphi_1 \implies \varphi_2$ then $\mathcal{M}_b(\varphi_1) \implies \mathcal{M}_b(\varphi_2)$.*

Disjunctive form. The monotone overapproximation can be related to a DNF representation of the original formula, a fact that we use extensively in §6 and also when we analyze Λ-PDR on specific examples. Starting with a DNF representation of φ, we can derive a DNF representation of $\mathcal{M}_b(\varphi)$ by dropping in each term the literals that agree with b. Intuitively, a "constraint" that $\sigma \models \ell$ in order to have $\sigma \models \mathcal{M}_b(t)$ where ℓ agrees with b is dropped because if $\sigma \models \mathcal{M}_b(t)$ then flipping a bit σ to disagree with b results in a state $\tilde{\sigma}$ such that also $\tilde{\sigma} \models \mathcal{M}_b(t)$, as $\sigma \leq_b \tilde{\sigma}$.

Lemma 4.5 (Generalization of Bshouty [1995], Lemma 1(7)). *Let $\varphi = t_1 \vee \ldots \vee t_m$ in DNF. Then the monotonization $\mathcal{M}_b(\varphi) \equiv \mathcal{M}_b(t_1) \vee \ldots \vee \mathcal{M}_b(t_m)$ where $\mathcal{M}_b(t_i) \equiv t_i \setminus b = \bigwedge\{\ell \in t_i \wedge \ell \notin b\}$.*

A corollary provides a canonical (inefficient) disjunctive form for $\mathcal{M}_b(\varphi)$:

Corollary 4.6. *Given a state v, we denote $cube_b(v) \stackrel{\text{def}}{=} \mathcal{M}_b(v) = \bigwedge\{p \mid v[p] = true,\ p \notin b\} \wedge \bigwedge\{\neg p_i \mid v[p] = false,\ \neg p \notin b\}$. Then $\mathcal{M}_b(\varphi) \equiv \bigvee_{v \models \varphi} cube_b(v)$.*

4.2 Monotone Hull

We now define the monotone hull, which is a conjunction of b-monotone overapproximations over all b from a fixed set of states B (in the case of Λ-PDR, $B = \mathcal{B}_k$). We start with the definition that uses a conjunction over monotone-overapproximations w.r.t. individual states, and then extend this to the union of (partial) cubes.

Definition 4.7 (Monotone Hull). The *monotone hull* of a formula φ w.r.t. a set of states B is $\mathrm{MHull}_B(\varphi) = \bigwedge_{b \in B} \mathcal{M}_b(\varphi)$.

The monotone hull can be simplified to use a succinct DNF representation of the basis B instead of a conjunction over all states. (This is the motivation for generalizing $\mathcal{M}_b(\cdot)$ to a cube b in §4.1.)

Lemma 4.8. *If $B = b_1 \vee \ldots \vee b_m$ and $b_1, \ldots, b_m$ are cubes, then* $\mathrm{MHull}_B(\varphi) \equiv \mathcal{M}_{b_1}(\varphi) \wedge \ldots \wedge \mathcal{M}_{b_m}(\varphi)$.

Note that when $B = b$ is a single cube, $\mathrm{MHull}_b(\varphi) = \mathcal{M}_b(\varphi)$.

Our main technical observation, connecting the monotone hull to Alg. 2, is that the monotone hull has an equivalent CNF form, as the conjunction of all overapproxmating clauses that exclude a state in B:

Theorem 4.9. $\mathrm{MHull}_B(\varphi) \equiv \bigwedge \{c \mid c \text{ is a clause}, \varphi \implies c, \text{ and } \exists b \in B.\, b \not\models c\}$.

Proof. $\implies$: Let c be a clause as in the rhs. Then there exists $b \in B$ s.t. $b \not\models c$. It suffices to show that $\mathcal{M}_b(\varphi) \implies c$. Recall that c is a disjunction of literals; since $b \not\models c$, all those literals are falsified in b. Hence, by Lemma 4.5, $\mathcal{M}_b(c) \equiv c$. Now $\varphi \implies c$ (by the choice of c), and Lemma 4.4 yields $\mathcal{M}_b(\varphi) \implies \mathcal{M}_b(c) \equiv c$.
$\impliedby$: Let σ be a model of the rhs. We want to prove that $\sigma \models \mathcal{M}_b(\varphi)$ for every $b \in B$. Assume otherwise. Take d as the conjunction of all literals that hold in both σ and b (if this set is empty, $d = \mathit{true}$). Clearly d is a term and $b, \sigma \models d$. Take $c = \neg d$; then c is a clause and $b \not\models c$. It remains to show that $\varphi \implies c$, because then c belongs to the rhs but $\sigma \not\models c$, in contradiction to the premise. To see this, note that $c = \mathcal{M}_b(\neg\sigma)$ by Lemma 4.5—all the literals from the clause $\neg\sigma$ on which b disagrees (because b agrees on them with σ). $\varphi \implies \neg\sigma$ because $\sigma \not\models \varphi$ (as $\sigma \not\models \mathcal{M}_b(\varphi)$, which is an overapproximation of φ). Hence Lemma 4.4 implies $\mathcal{M}_b(\varphi) \implies \mathcal{M}_b(\neg\sigma) = c$, and in particular $\varphi \implies c$. The claim follows. □

We can now derive Equation (2):

Corollary 4.10. *In Λ-PDR (Alg. 2), $\mathcal{F}_{i+1} = \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i))$.*

Proof. $\mathcal{F}_{i+1}$ is the conjunction formed by the process that for each $\sigma_b \in \mathcal{B}_k$ (line 11 of Alg. 2) iterates in line 12 over all clauses that exclude σ_b, and conjoins c if it overapproximates $\underline{\delta}(\mathcal{F}_i)$ (line 13). By Thm. 4.9 this is $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i))$. □

Example 4.11. The above lemmas are the basis for our presentation in §2.4 of $\mathcal{F}_1$ on Fig. 1 as Equation (3). Let us now use these lemmas to describe later frames in that execution. Recall that $\mathcal{B}_k$ is the cube $\overline{x} = 10\ldots0 \wedge y_n y_{n-1} \ldots y_1 = 11\ldots1 \wedge z = 1$, denote it by b. For the next frame, $\underline{\delta}(\mathcal{F}_1) = \mathcal{F}_1 \vee (\overline{x} = 10\ldots01 \wedge \overline{y} = 0\ldots0 \wedge z = 0)$. Then

$$\begin{aligned}\mathcal{F}_2 &\underset{\text{Thm. 4.9}}{=} \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_1)) \underset{\text{Lemma 4.8}}{=} \mathcal{M}_b(\underline{\delta}(\mathcal{F}_1)) \\ &\underset{\text{Lemma 4.5}}{=} \mathcal{M}_b(\mathcal{F}_1) \vee \mathcal{M}_b(\overline{x} = 10\ldots01 \wedge \overline{y} = 0\ldots0 \wedge z = 0) \\ &\underset{\text{Lemma 4.5}}{=} \mathcal{F}_1 \vee (x_0 = 1 \wedge \overline{y} = 0\ldots0 \wedge z = 0).\end{aligned}$$

For the next frame, $\underline{\delta}(\mathcal{F}_2) = \mathcal{F}_2 \vee (x_0 = 0 \wedge \overline{y} = 0\ldots0 \wedge z = 0) \wedge \neg(\overline{x} = 1\ldots0 \wedge \overline{y} = 0\ldots0 \wedge z = 0)$. This is equivalent to $(\overline{y} = 0\ldots0 \wedge z = 0) \wedge \neg(\overline{x} = 1\ldots0 \wedge \overline{y} = 0\ldots0 \wedge z = 0)$, which is already

b-monotone and hence this is also $\mathcal{F}_3 = \mathrm{MHull}_b(\underline{\delta}(\mathcal{F}_2))$. $\underline{\delta}(\mathcal{F}_3) = \mathcal{F}_3$, and so $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_3)) = \mathcal{F}_3$ (see Lemma 4.15 below), and the algorithm converges.

Example 4.12. In the previous example, $\mathcal{B}_k$ consisted of a single cube. To exemplify the more general case, consider a system over n variables $x_1, \ldots, x_n$, with $\mathit{Init} = (x_1 = \ldots = x_n = 0)$, Bad the set of states with exactly one variable 1, and δ that non-deterministically chooses some $i \neq j$ with $x_i = x_j = 0$ and sets $x_i \leftarrow 1$ and $x_j \leftarrow 1$. Take $k = 0$. Then $\mathcal{B}_k = b_1 \vee \ldots \vee b_n$ where b_i is $x_i = 1 \wedge \bigwedge_{j \neq i} (x_j = 0)$. After one step, $\underline{\delta}(\mathcal{F}_0) = (0 \ldots 0) \vee \bigvee_{i_1 \neq i_2} (x_{i_1} = x_{i_2} = 1 \wedge \bigwedge_{j \notin \{i_1, i_2\}} (x_j = 0))$ is the set of states where there are zero or two variables 1. Then for every b_i,

$$\mathcal{M}_{b_i}(\underline{\delta}(\mathcal{F}_0)) \underset{\text{Lemma 4.5}}{=} (x_i = 1) \vee \left(\bigvee_{i_1 \neq i_2, i \notin \{i_1, i_2\}} x_{i_1} = x_{i_2} = x_i = 1 \right) \vee \left(\bigvee_{i_1 \neq i_2, i = i_1} x_{i_2} = 1 \right) = \bigvee_j (x_j = 1).$$

Hence, $\mathcal{F}_1 \underset{\text{Thm. 4.9}}{=} \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_0)) = \bigvee_i \mathcal{M}_{b_i}(\underline{\delta}(\mathcal{F}_0)) = \bigvee_j (x_j = 1)$. After another step, $\underline{\delta}(\mathcal{F}_1) = \mathcal{F}_1$ and so $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_1)) = \mathcal{F}_1$ (see Lemma 4.15 below), and the algorithm converges.

Additional lemmas. Before proceeding, we state a few helpful, simple lemmas that we use later:

Lemma 4.13. $\varphi \Longrightarrow \mathrm{MHull}_B(\varphi)$.

Lemma 4.14. *If* $\varphi_1 \Longrightarrow \varphi_2$ *then* $\mathrm{MHull}_b(\varphi_1) \Longrightarrow \mathrm{MHull}_b(\varphi_2)$.

Lemma 4.15. *The monotone hull is idempotent, that is,* $\mathrm{MHull}_B(\mathrm{MHull}_B(\varphi)) \equiv \mathrm{MHull}_B(\varphi)$.

5 ABSTRACT INTERPRETATION IN THE MONOTONE SPAN OF $\mathcal{B}_k$

In this section, we cast Λ-PDR as an abstract interpretation in a new logical abstract domain of the *monotone span of* $\mathcal{B}_k$. We first discuss abstract interpretation in general, and then develop the notion of a monotone span. We then define the abstract domain and show the connection to Λ-PDR.

5.1 Background: Abstract Interpretation

In this section, we review the basics of abstract interpretation [Cousot and Cousot 1977]; see [Rival and Yi 2020; Urban 2015] for complete and general presentations. A *complete join-semilattice* is a tuple $\langle D, \sqsubseteq, \sqcup, \bot \rangle$ where D is partially-ordered by $\sqsubseteq$, $\bigsqcup X$ is the least upper bound of every $X \subseteq D$ ($\forall x \in X.\ x \sqsubseteq \bigsqcup X$ and X is the smallest w.r.t. $\sqsubseteq$ that satisfies this), and $\bot$ is the minimal element ($\forall x \in D.\ \bot \sqsubseteq x$). A *chain* is a sequence of elements from D satisfying $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$, and a *strictly ascending chain* if additionally $x_i \neq x_{i+1}$ for every i. The lattice's *height* is the maximal length of a strictly ascending chain. We consider finite domains, where in particular the height is also finite.

A function $F : D \to D$ is Scott-continuous if for every chain $X = x_0, x_1, \ldots \subseteq D$ it holds $f(\bigsqcup_{x \in X} x) = \bigsqcup_{x \in X} f(x)$. By the Knaster-Tarski theorem, such F has a least fixed-point (lfp)—the least x such that $f(x) = x$—and by Kleene's theorem it is $\bigsqcup_{i \geq 0} F^i(\bot)$. When the domain's height is also finite, the sequence $\{F^i(\bot)\}_{i \geq 0}$ converges to the lfp.

In our setting, the *concrete domain* is the join-semilattice powerset domain of the set of states, $C = \langle D = 2^{\mathrm{States}[\Sigma]}, \subseteq, \cup, \emptyset \rangle$. An *abstract domain* is a join-semilattice $\mathcal{A} = \langle D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \bot^\sharp \rangle$. An *abstraction function* is a monotone function $\alpha : D \to D^\sharp$, that is, $\forall S_1 \subseteq S_2 \in D.\ \alpha(S_1) \sqsubseteq^\sharp \alpha(S_2)$. A *conretization function* is a monotone function $\gamma : D^\sharp \to D$, that is, $\forall a_1 \sqsubseteq^\sharp a_2 \in D^\sharp.\ \gamma(a_1) \subseteq \gamma(a_2)$. There is a *Galois connections* between $(D, \subseteq)$ and $(D^\sharp, \sqsubseteq^\sharp)$ through (α, γ), denoted $(D, \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (D^\sharp, \sqsubseteq^\sharp)$, if $\forall S \in D, a \in D^\sharp.\ \alpha(S) \sqsubseteq^\sharp a \Leftrightarrow S \subseteq \gamma(a)$.

Let (Init, δ) be a transition system. Define the concrete transformer $F_{\mathit{Init},\delta} : D \to D$ by $F_{\mathit{Init},\delta}(S) = \delta(S) \cup \mathit{Init}$. It is Scott-continuous, since for every increasing $\subseteq$-chain $X \subseteq D$, we have $F_{\mathit{Init},\delta}(\bigcup_{S \in X} S) = \bigcup_{S \in X} F_{\mathit{Init},\delta}(S)$. Its fixed-point $\mathit{lfp}(F_{\mathit{Init},\delta})$ is the set of reachable states of (Init, δ). The corresponding

best abstract transformer is given by $F^\sharp_{Init,\delta}(a) = \alpha(F_{Init,\delta}(\gamma(a)))$. $F^\sharp_{Init,\delta}$ is also Scott-continuous, and there is a *fixed-point transfer* from F to $F^\sharp$: $lfp(F^\sharp_{Init,\delta}) = \alpha(lfp(F_{Init,\delta}))$. It follows that $lfp(F^\sharp_{Init,\delta})$ is the least abstraction of the set of reachable states, and it is obtained by the chain

$$\bot^\sharp \sqsubseteq^\sharp (F^\sharp_{Init,\delta})^1(\bot^\sharp) \sqsubseteq^\sharp (F^\sharp_{Init,\delta})^2(\bot^\sharp) \sqsubseteq^\sharp \ldots$$

at its convergence point in a finite i^* with $(F^\sharp_{Init,\delta})^{i^*} = (F^\sharp_{Init,\delta})^{i^*+1}$ (due to the finite height of $D^\sharp$). We call the chain the *Kleene iterations with the best transformer*, and overall it converges to the most precise sound inductive invariant in the abstract domain.

Another way to phrase the same chain, denoting $\xi_i = (F^\sharp_{Init,\delta})^i(\bot^\sharp)$, is by

$$\xi_1 = \alpha(Init) \qquad\qquad \xi_{i+1} = \xi_i \sqcup^\sharp \alpha(\delta(\gamma(\xi_i)) = \alpha(\underline{\delta}(\gamma(\xi_i))),$$

because $\xi_i \sqsubseteq^\sharp \xi_{i+1}$ implies that $\gamma(\xi_i) \subseteq \gamma(\xi_{i+1})$ and therefore $\xi_{i+1} = F^\sharp_{Init,\delta}(\xi_i) = F^\sharp_{Init,\delta}(\xi_i) \sqcup^\sharp \alpha(\delta(\gamma(\xi_i)) \cup Init) = \alpha(\delta(\gamma(\xi_i)) \cup Init \cup \gamma(\xi_i)) = \alpha(\underline{\delta}(\gamma(\xi_i)))$.

5.2 Monotone Basis and Monotone Span

We define the abstract domain in which Λ-PDR operates using the notion of a monotone span.

Definition 5.1 (Monotone Basis [Bshouty 1995]). A *monotone basis* is a set of states B. It is a basis *for a formula* φ if $\varphi \equiv \mathrm{MHull}_B(\varphi)$.

Definition 5.2 (Monotone Span). $\mathrm{MSpan}(B) = \{\mathrm{MHull}_B(\varphi) \mid \varphi \text{ over } \Sigma\}$, the set of formulas for which B is a monotone basis.

Bshouty [1995] showed that $\varphi \in \mathrm{MSpan}(B)$ iff there exist clauses $c_1, \ldots, c_s$ such that $\varphi \equiv c_1 \wedge \ldots \wedge c_s$ and for every $1 \le i \le s$ there exists $b_j \in B$ such that $b_j \not\models c_i$. (This is also a corollary of Thm. 4.9.) The monotone span is thus the set of all formulas that can be written in CNF using clauses that exclude states from the basis. A consequence of this is that it is closed under conjunction:

Lemma 5.3. *If* $\varphi_1, \varphi_2 \in \mathrm{MSpan}(B)$ *then also* $\varphi_1 \wedge \varphi_2 \in \mathrm{MSpan}(B)$.

Proof. By Thm. 4.9, $\varphi_1 \equiv \bigwedge \{c \mid c \text{ is a clause}, \varphi_1 \Longrightarrow c \text{ and } \exists b \in B.\, b \not\models c\}$, likewise for φ_2. The set $\{c \mid c \text{ is a clause}, \varphi_1 \wedge \varphi_2 \Longrightarrow c, \text{ and } \exists b \in B.\, b \not\models c\}$ includes the conjuncts associated with both φ_1, φ_2 (and more), and so $\mathrm{MHull}_B(\varphi_1 \wedge \varphi_2) \Longrightarrow \varphi_1 \wedge \varphi_2$. The other implication is by Lemma 4.13. □

5.3 Abstract Interpretation in the Monotone Span

For a set of states B, we define the abstract domain $\mathbb{M}[B] = \langle \mathrm{MSpan}(B), \Longrightarrow, \sqcup_B, false\rangle$, a logical abstract domain [Gulwani et al. 2008] consisting of the set of (propositional) formulas for which B is a monotone basis (Def. 5.1), ordered by logical implication, with bottom element *false*. The existence of $\sqcup_B$ relies on Lemma 5.3 (the least upper-bound is the conjunction of all upper-bounds), and $false \in \mathrm{MSpan}(B)$ because $\mathrm{MHull}_B(false) = false$, seeing that $\mathcal{M}_b(false) = false$ for every b.

To define a Galois connection [Cousot and Cousot 1977] between sets of concrete states and formulas in $\mathrm{MSpan}(B)$, we use the *concretization* $\gamma(\varphi) = \{\sigma \mid \sigma \models \varphi\}$ (in the sequel, we refer to γ as the identity function, by our convention of equating formulas with the set of states they represent). The best abstraction is expressed by $\alpha_B(S) = \mathrm{MHull}_B(S)$:

Lemma 5.4. *Let* $S \subseteq States[\Sigma]$. *Then* $\mathrm{MHull}_B(S)$ *is the least overapproximation of* S *in* $\mathrm{MSpan}(B)$, *namely,* $\mathrm{MHull}_B(S) \Longrightarrow \varphi$ *for every* $\varphi \in \mathrm{MSpan}(B)$ *s.t.* $S \Longrightarrow \varphi$.

Proof. Since $\varphi \in \mathrm{MSpan}(B)$, by Thm. 4.9, $\varphi \equiv \bigwedge_i c_i$ where each clause c_i excludes some $a_i \in B$. If $\alpha_k(S) \not\Longrightarrow \varphi$, then there is some c_i and a corresponding $a_i \in B$ that it excludes such that

$\alpha_B(S) \not\Longrightarrow c_i$. This means that $\mathcal{M}_b(S) \not\Longrightarrow c_i$ for every $b \in B$. But then in particular $\mathcal{M}_{a_i}(S) \not\Longrightarrow c_i$, even though $S \Longrightarrow c_i$ and $a_i \not\models c_i$, which is a contradiction to Corollary 4.6 for $\mathcal{M}_{a_i}(S)$. □

Lemma 5.5. *There is a Galois connection* $(2^{States[\Sigma]}, \subseteq) \underset{\alpha_B}{\overset{\gamma}{\leftrightarrows}} (\mathrm{MSpan}(B), \Longrightarrow)$.[6]

Proof. α_B is monotone by Lemma 4.14. γ is also monotone. Let $\varphi \in \mathrm{MSpan}(B)$ and $S \subseteq States[\Sigma]$. If $\alpha_B(S) \Longrightarrow \varphi$ then $S \subseteq \gamma(\varphi)$ since $S \subseteq \mathrm{MHull}_B(S)$ (Lemma 4.13). If $S \subseteq \gamma(\varphi)$ then $\alpha_B(S) \Longrightarrow \varphi$ by Lemma 5.4. □

Remark 5.1 (Disjunctive completion). *When $B = b_1 \vee \ldots \vee b_m$ is a disjunction of multiple cubes, the domain is not disjunctively-complete [Cousot and Cousot 1979]: if $\varphi_1, \varphi_2 \in \mathrm{MSpan}(B)$, it could be that $\varphi_1 \vee \varphi_2 \notin \mathrm{MSpan}(B)$. However, for a single cube b, the join operation of $\mathbb{M}[b]$ is disjunction, as follows from Lemma 4.5. In this case, a definition of the abstraction α_b through the representation function $\beta_b(\sigma) = cube_b(\sigma)$ is straightforward, since $\alpha_b(S) = \bigsqcup_{\sigma \in S} \beta_b(\sigma)$ reads as Corollary 4.6.*

Remark 5.2 (As a reduced product domain). *One way to understand $\mathbb{M}[B]$ when $B = b_1 \vee \ldots \vee b_m$ is as a reduced product [Cousot and Cousot 1979] of the per-cube domains: $\mathbb{M}[B]$ (quotient on logical equivalence) is isomorphic to $\bigotimes_i \mathbb{M}[b_i]$. The Cartesian product domain is over m-tuples of formulas, $\bigtimes_i \mathrm{MSpan}(b_i)$, ordered by $(\varphi_1^1, \ldots, \varphi_m^1) \sqsubseteq (\varphi_1^2, \ldots, \varphi_m^2) \iff \bigwedge_{i=1}^m (\varphi_i^1 \Longrightarrow \varphi_i^2)$, with concretization $\gamma_{\times b_i}(\varphi_1, \ldots, \varphi_m) = \bigcap_{i=1}^m \gamma(\varphi_i)$. The reduced product quotients the Cartesian product w.r.t. having the same concretization. This is isomorphic to $\mathbb{M}[B]$ because $\gamma_{\times b_i} = \gamma(\bigwedge_{i=1}^m \varphi_i)$, and if $\varphi_i \in \mathrm{MSpan}(b_i)$ then $\bigwedge_{i=1}^m \varphi_i \in \mathrm{MSpan}(B)$.*

Λ-PDR as Kleene iterations. Alg. 3 shows Kleene iterations in $\mathbb{M}[\mathcal{B}_k]$ with the best abstract transformer for $(Init, \delta)$. The next iterate (line 6) is always $\mathcal{F}_{i+1}^{\mathrm{ai}} = \alpha_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i^{\mathrm{ai}}))$, which exactly matches the relation between successive frames in Λ-PDR (Corollary 4.10). This means that Λ-PDR's frames exactly match the Kleene iterates, at least when the initial states are in $\mathrm{MSpan}(\mathcal{B}_k)$ themselves (in which case the first iterate in line 4 is simply *Init*); otherwise there is a difference of at most one frame:

Algorithm 3 Kleene Iterations in $\mathbb{M}[\mathcal{B}_k]$

1: **procedure** Λ-PDR($Init, \delta, Bad, k$)
2: $i \leftarrow 0$
3: $\mathcal{F}_{-1}^{\mathrm{ai}} \leftarrow false$
4: $\mathcal{F}_0^{\mathrm{ai}} \leftarrow \alpha_{\mathcal{B}_k}(Init)$
5: **while** $\mathcal{F}_i^{\mathrm{ai}} \not\Longrightarrow \mathcal{F}_{i-1}^{\mathrm{ai}}$ **do**
6: $\mathcal{F}_{i+1}^{\mathrm{ai}} = \alpha_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i^{\mathrm{ai}}))$
7: $i \leftarrow i + 1$
8: **return** $\mathcal{F}_i^{\mathrm{ai}}$

Theorem 5.6. *$\mathcal{F}_i^{ai} \subseteq \mathcal{F}_{i+1} \subseteq \mathcal{F}_{i+1}^{ai}$ for every i where $\mathcal{F}_{i+1}$ exists. Further, if $Init \in \mathrm{MSpan}(\mathcal{B}_k)$, then $\mathcal{F}_i = \mathcal{F}_i^{ai}$.*

Proof. By induction on i, first prove that $\mathcal{F}_i \subseteq \mathcal{F}_i^{\mathrm{ai}}$ for every i where $\mathcal{F}_i$ exists. Initially, $\mathcal{F}_0 = Init \subseteq \alpha_{\mathcal{B}_k}(Init) = \mathcal{F}_0^{\mathrm{ai}}$. For the step, assume that $\mathcal{F}_i \subseteq \mathcal{F}_i^{\mathrm{ai}}$. Using Lemma 4.14, this implies that $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i)) \subseteq \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i^{\mathrm{ai}}))$, which by Corollary 4.10 and the Kleene iterations means that $\mathcal{F}_{i+1} \subseteq \mathcal{F}_{i+1}^{\mathrm{ai}}$. For the other inclusion, likewise, since $\mathcal{F}_0^{\mathrm{ai}} = \alpha_{\mathcal{B}_k}(Init) = \mathrm{MHull}_{\mathcal{B}_k}(Init) \subseteq \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(Init)) = \mathcal{F}_1$, we have by induction that $\mathcal{F}_i^{\mathrm{ai}} \subseteq \mathcal{F}_{i+1}$ for every i s.t. $\mathcal{F}_{i+1}$ exists. Similarly, if $Init \in \mathrm{MSpan}(\mathcal{B}_k)$ then $\alpha_{\mathcal{B}_k}(Init) = Init$, and by induction $\mathcal{F}_i = \mathcal{F}_i^{\mathrm{ai}}$ for every i. □

We can now relate the number of frames in Λ-PDR and the number of Kleene iterations:

Corollary 5.7. *Λ-PDR($Init, \delta, Bad, k$) (Alg. 2) converges or fails (line 8) in a frame whose index is at most one greater than the number of Kleene iterations in $\mathbb{M}[\mathcal{B}_k]$ on $(Init, \delta)$ (Alg. 3).*

Proof. Let i^* be the iteration where Alg. 3 converges to the least-fixed point, i.e. $\mathcal{F}_{i^*+1}^{\mathrm{ai}} = \mathcal{F}_{i^*}^{\mathrm{ai}}$. If Alg. 2 does not terminate with error (line 8) before $i = i^* + 1$, by Thm. 5.6 $\mathcal{F}_i^{\mathrm{ai}} \subseteq \mathcal{F}_{i+1} \subseteq \mathcal{F}_{i+1}^{\mathrm{ai}}$ for every $i \leq i^*$, and for $i = i^*$ we have $\mathcal{F}_{i^*}^{\mathrm{ai}} \subseteq \mathcal{F}_{i+1} \subseteq \mathcal{F}_{i^*+1}^{\mathrm{ai}} = \mathcal{F}_{i^*}^{\mathrm{ai}}$, so $\mathcal{F}_{i^*+1}$ is an inductive invariant.

[6] Quotient over logical equivalence, this is a Galois *insertion*, as $\alpha_B(\gamma(\psi)) = \mathrm{MHull}_B(\psi) \equiv \psi$ for $\psi \in \mathrm{MSpan}(B)$.

Therefore, $\underline{\delta}(\mathcal{F}_{i^*+1}) = \mathcal{F}_{i^*+1}$, and thus, because $\mathcal{F}_{i^*+1} \in \mathrm{MSpan}(\mathcal{B}_k)$, also $\mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_{i^*+1})) = \mathcal{F}_{i^*+1}$ and the algorithm converges. □

Further, Λ-PDR converges whenever the Kleene iterations converge to an inductive invariant:

Corollary 5.8. *If there exists an inductive invariant $I \in \mathrm{MSpan}(\mathcal{B}_k)$ for $(Init, \delta, Bad)$, then Λ-PDR$(Init, \delta, Bad, k)$ (Alg. 2) converges to an inductive invariant.*

Proof. Alg. 3 converges from below to the abstract lfp for $(Init, \delta)$ in $\mathbb{M}[\mathcal{B}_k]$, which from the premise is strong enough to prove safety. Using Thm. 5.6, the same is true for Alg. 2. □

Increasing k refines $\mathbb{M}[\mathcal{B}_k]$. Increasing the backward exploration bound k refines $\mathbb{M}[\mathcal{B}_k]$ by increasing $\mathrm{MSpan}(\mathcal{B}_k)$ ($\mathcal{B}_k \subsetneq \mathcal{B}_{k'}$ implies $\mathrm{MSpan}(\mathcal{B}_k) \subsetneq \mathrm{MSpan}(\mathcal{B}_{k'})$: every $\varphi \in \mathrm{MSpan}(\mathcal{B}_k)$ is also $\varphi \in \mathrm{MSpan}(\mathcal{B}_{k'})$, and for instance the clause $\neg\sigma_b \in \mathrm{MSpan}(\mathcal{B}_{k'}) \setminus \mathrm{MSpan}(\mathcal{B}_k)$ if the state $\sigma_b \in \mathcal{B}_{k'} \setminus \mathcal{B}_k$). Restarting Λ-PDR with a larger k (line 9 in Alg. 2) thus refines the domain until it includes an inductive invariant that establishes safety. Such a k always exists because the set of all backward reachable states (attained by some finite k in the setting of propositional systems) is sufficient to express the weakest safe inductive invariant.

Efficient convergence. Unfortunately, the lattice height of $\mathrm{MSpan}(B)$ is exponential. For example, all the formulas in the strictly ascending chain of formulas $\{\overline{x} \le i\}_{0 \le i \le 2^n-1}$ over n variables $\overline{x} = x_{n-1}, \ldots, x_0$ are in $\mathrm{MSpan}(\{0 \ldots 0\})$ (see §9). Therefore, to bound the number of iterations we need to consider properties of the transition system, a task on which we embark next.

6 CONVERGENCE BOUNDS VIA ABSTRACT DIAMETER

In this section and the next, we prove a bound on the number of iterations of Λ-PDR on a given transition system via the DNF size of a monotonization of the transition relation. In the current section we assume that $\mathcal{B}_k$ can be expressed as a single cube b, and generalize it in §7.

To formulate the bound, we define a monotonization of the transition relation, which is a formula over $\Sigma \cup \Sigma'$. For cubes c_1 and c_2 over Σ, we denote by $\mathcal{M}_{(c_1,c_2)}(\delta)$ the monotonization $\mathcal{M}_a(\delta)$ where $a = c_1 \wedge c_2'$ and c_2' is obtained from c_2 by substituting each $p \in \Sigma$ by the corresponding $p' \in \Sigma'$.

The monotonization we perform on the pre-state vocabulary Σ uses the *reflection* of the backward reachable cube:

Definition 6.1 (Reflection). For a cube $b = \ell_1 \wedge \ldots \wedge \ell_r$, the *reflection* is $Ref(b) = \neg\ell_1 \wedge \ldots \wedge \neg\ell_r$.

Our main theorem in this section is as follows.

Theorem 6.2. *Let $(Init, \delta, Bad)$ be a transition system, and $\mathcal{B}_k = b$ a cube. Then Λ-PDR$(Init, \delta, Bad, k)$ converges or fails in a frame whose index is bounded by $\left|\mathcal{M}_{(Ref(b),b)}(\delta)\right|_{\mathrm{dnf}} + 1$.*

To prove this we construct an "abstract" transition system whose diameter is the number of iterations required for Alg. 3 to converge (§6.1), and use it to derive sufficient conditions for rapid convergence of Λ-PDR (§6.2). Throughout, we fix a transition system $(Init, \delta, Bad)$ and a backwards bound $k \in \mathbb{N}$, denoting the cube $\mathcal{B}_k$ by b.

6.1 Abstract Transition System

Given $(Init, \delta, Bad)$, the *abstract transition system* $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ is defined over the same set of states as the original, and extends its transitions:

Definition 6.3 (Abstract Transition System). The abstract transition system of $(Init, \delta, Bad)$ w.r.t. b is defined as a transition system over States$[\Sigma]$ with $Init^\sharp = \mathcal{M}_b(Init)$, $Bad^\sharp = Bad$, and $\delta^\sharp =$

$\mathcal{M}_{(true,b)}(\delta)$. More explicitly (through Corollary 4.6),

$$(\sigma, \sigma') \in \delta^\sharp \iff \exists \sigma''.\ (\sigma, \sigma'') \in \delta \wedge \sigma' \in cube_b(\sigma'').$$

A step of the abstract transition system is composed of a step of the original one and a b-monotone overapproximation from a "protector" state σ'' (see §2.4). This mimics a step of the algorithm, which computes the post-image and then a monotone overapproximation. A potentially critical difference is that the transition system performs this state-by-state, while the algorithm computes these operations over sets. The insight is that when $\mathcal{B}_k$ is a single cube, the abstraction of Λ-PDR factors to individual states as well, and so can be captured using an ordinary transition system.

Lemma 6.4. *Let $R_i^\sharp$ be the set of states reachable in $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ w.r.t. $\mathcal{B}_k = b$ (Def. 6.3) in at most i steps. Then $R_i^\sharp \equiv \mathcal{F}_i^{ai}$ where $\mathcal{F}_i^{ai}$ is the i'th iterate of Alg. 3 in $\mathbb{M}[b]$ on $(Init, \delta, Bad)$.*

Proof. By induction on i. Initially, $R_0^\sharp = Init^\sharp = \mathcal{M}_b(Init) = \mathcal{F}_0^{\mathrm{ai}}$. For the step, by the definition of the abstract system, by the induction hypothesis $R_i^\sharp \equiv \mathcal{F}_i^{\mathrm{ai}} \in \mathrm{MSpan}(b)$. Hence,

$$R_{i+1}^\sharp = \underline{\delta^\sharp}(R_i^\sharp) = R_i^\sharp \cup \bigvee_{\sigma'' \in \delta(R_i^\sharp)} cube_b(\sigma'') \underset{\text{Corollary 4.6}}{=} R_i^\sharp \cup \mathcal{M}_b(\delta(R_i^\sharp))$$

$$\underset{R_i^\sharp \in \mathrm{MSpan}(b)}{=} \mathcal{M}_b(R_i^\sharp) \cup \mathcal{M}_b(\delta(R_i^\sharp)) \underset{\text{Lemma 4.5}}{=} \mathcal{M}_b(\underline{\delta}(R_i^\sharp)) \underset{\text{ind.}}{=} \mathcal{M}_b(\underline{\delta}(\mathcal{F}_i^{\mathrm{ai}})) = \mathcal{F}_{i+1}^{\mathrm{ai}}. \qquad \square$$

Corollary 6.5. *Let $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ be the abstract transition system w.r.t. $\mathcal{B}_k = b$ (Def. 6.3). If $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ is safe and its reachability diameter is s, then Λ-PDR$(Init, \delta, Bad, k)$ converges in frame at most $s + 1$. If $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ reaches a bad state in s steps, then Λ-PDR$(Init, \delta, Bad, k)$ fails (line 8) in frame at most $s + 1$.*

Proof. From Lemma 6.4, $\mathcal{F}_s^{\mathrm{ai}} \equiv \mathcal{F}_{s+1}^{\mathrm{ai}}$ iff $R_s^\sharp \equiv R_{s+1}^\sharp$, and the least s in which the latter holds is the diameter. For the unsafe case, $\mathcal{F}_s^{\mathrm{ai}} \cap Bad \neq \emptyset$ iff $R_s^\sharp \cap Bad \neq \emptyset$. Apply Corollary 5.7 in both cases to deduce convergence, resp. failure, of Λ-PDR in frame at most $s + 1$. $\square$

6.2 Diameter Bound via Abstract DNF Size

In this section, we bound the diameter of the abstract transition system as follows:

Lemma 6.6. *Let $(Init, \delta, Bad)$ be a transition system, and $\mathcal{B}_k = b$. Then the reachability diameter of $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ is at most $\left|\mathcal{M}_{(Ref(b),b)}(\delta)\right|_{\mathrm{dnf}}$.*

The proof uses a simple, general bound on the diameter of transition systems:

Lemma 6.7. *The reachability diameter of a transition system $(Init, \delta, Bad)$ is bounded by $|\delta|_{\mathrm{dnf}}$.*

Proof. Fix a minimal DNF representation of δ. Thinking about each disjunct of δ as an action a, every transition can be labeled by at least one action. Whenever in an execution $\sigma_1, \sigma_2, \ldots$ an action a labels two transitions $\sigma_{i_1} \xrightarrow{a} \sigma_{i_1+1}, \sigma_{i_2} \xrightarrow{a} \sigma_{i_2+1}$, the segment between the occurrences, $\sigma_{i_1+1}, \ldots, \sigma_{i_2}$ can be dropped and the resulting trace is still valid (and terminates at the same state)—this is because if $(\sigma_{i_1}, \sigma_{i_1+1}) \models a$ and likewise $(\sigma_{i_2}, \sigma_{i_2+1}) \models a$ then also $(\sigma_{i_1}, \sigma_{i_2+1}) \models a$, because a, which is a cube, can be decomposed to $a_{pre} \wedge a_{post}$ where all the literals in a_{pre} are in Σ and those in a_{post} are in Σ'. Overall, every state that can be reached from another state can do so by an execution where each action appears at most once, and thus the diameter is bounded by $|\delta|_{\mathrm{dnf}}$. $\square$

We apply Lemma 6.7 to a further abstracted transition relation, $\delta^{\natural} = \mathcal{M}_{(Ref(b),b)}(\delta)$, in which the pre-state vocabulary also undergoes monotonization, whose diameter we relate to that of $\delta^{\sharp} = \mathcal{M}_{(true,b)}(\delta)$ (although $\delta^{\sharp}, \delta^{\natural}$ are not bisimilar). The switch from $\delta^{\sharp}$ to $\delta^{\natural}$ can improve the resulting bound, and never worsens it, because monotonization never increases DNF size (a corollary of Lemma 4.8), so $\left|\delta^{\natural}\right|_{\text{dnf}} = \left|\mathcal{M}_{(Ref(b),true)}(\delta^{\sharp})\right|_{\text{dnf}} \leq \left|\delta^{\sharp}\right|_{\text{dnf}}$. We start with a technical lemma about the monotonization w.r.t. a reflection.

Lemma 6.8. *$\sigma_1 \models cube_{Ref(b)}(\sigma_2) \iff \sigma_2 \models cube_b(\sigma_1)$ for every cube b and states σ_1, σ_2.*

Proof of Lemma 6.6. Consider the transition system $Init^{\natural} = \mathcal{M}_b(Init)$, $Bad^{\natural} = Bad$, and $\delta^{\natural} = \mathcal{M}_{(Ref(b),b)}(\delta)$. We claim that the reachability diameter of $(Init^{\natural}, \delta^{\natural}, Bad^{\natural})$ coincides with that of $(Init^{\sharp}, \delta^{\sharp}, Bad^{\sharp})$. Applying Lemma 6.7 to $(Init^{\natural}, \delta^{\natural}, Bad^{\natural})$ implies the desired bound.

First, $\delta^{\sharp} \subseteq \delta^{\natural}$. This is because if $(\sigma, \sigma') \in \delta^{\sharp}$, then by definition there is σ'' such that $(\sigma, \sigma'') \in \delta$ and $\sigma' \in cube_b(\sigma'')$. Considering the product monotone order, $(\sigma, \sigma'') \leq_{(\cdot,b)} (\sigma, \sigma')$, and so $(\sigma, \sigma'') \in \delta \implies (\sigma, \sigma') \in \mathcal{M}_{(Ref(b),b)}(\delta) = \delta^{\natural}$, as required.

Second, we show that for any $S \in \text{MSpan}(b)$ it holds that $\underline{\delta^{\natural}}(S) \subseteq \underline{\delta^{\sharp}}(S)$. $\underline{\delta^{\natural}}(S) = S \cup \delta^{\natural}(S)$ and $\underline{\delta^{\sharp}}(S) = S \cup \delta^{\sharp}(S)$, so we need to show that $\delta^{\natural}(S) \subseteq \delta^{\sharp}(S)$. Let $(\sigma, \sigma') \in \delta^{\natural}, \sigma \in S$. By the definition of $\delta^{\natural}$ and Corollary 4.6, there exists $(\widetilde{\sigma}, \widetilde{\sigma}') \in \delta$ such that $\sigma \models cube_{Ref(b)}(\widetilde{\sigma})$ and $\sigma' \models cube_b(\widetilde{\sigma}')$. The former implies, by Lemma 6.8, that $\widetilde{\sigma} \models cube_b(\sigma)$, hence $\widetilde{\sigma} \in S$ as well (because $S \in \text{MSpan}(b)$). Writing $\sigma'' = \widetilde{\sigma}'$ shows that $(\widetilde{\sigma}, \sigma') \in \delta^{\sharp}$, and hence $\sigma' \in \underline{\delta^{\sharp}}(S)$, as required.

Let $R_i^{\sharp}, R_i^{\natural}$ be the set of states reachable in at most i steps in $(Init^{\sharp}, \delta^{\sharp}, Bad^{\sharp}), (Init^{\natural}, \delta^{\natural}, Bad^{\natural})$ respectively. The first part of the argument (and induction on i) shows that $R_i^{\sharp} \subseteq R_i^{\natural}$. By Lemma 6.4, always $R_i^{\sharp} \in \text{MSpan}(b)$, and therefore the second argument above shows that $R_i^{\natural} \subseteq R_i^{\sharp}$. □

Combining the above results yields a proof of the main theorem of this section.

Proof of Thm. 6.2. By Lemma 6.4, the number of iterations before convergence or failure of Λ-PDR is bounded by 1 plus the reachability diameter of $(Init^{\sharp}, \delta^{\sharp}, Bad^{\sharp})$, which by Lemma 6.6 is at most $\left|\mathcal{M}_{(Ref(b),b)}(\delta)\right|_{\text{dnf}}$. □

Complexity. Finding whether there is an equivalent DNF representation with at most s terms is complete for the second level of the polynomial hierarchy Σ_2^P [Umans 2001]. This is on par with deciding whether the diameter is bounded by s [Hemaspaandra et al. 2010] (see also [Schaefer and Umans 2002]). Thus the bound in Thm. 6.2 is not an efficiently-computable upper bound on the number of frames of Λ-PDR. Instead, we view the result of Thm. 6.2 as a conceptual explanation of how smaller diameters can originate from the abstraction.

7 CONVERGENCE BOUNDS VIA ABSTRACT HYPERTRANSITION SYSTEMS

In this section, we generalize the results of §6 to the case that $\mathcal{B}_k$ is not expressible as a single cube. In this case, our bound is the product of monotonizations w.r.t. the different cubes that comprise $\mathcal{B}_k = b_1 \vee \ldots \vee b_m$ in the post-state, and w.r.t. (the reflection of) the least cube that contains all $\mathcal{B}_k$ in the pre-state, defined as follows:

Definition 7.1. If $\mathcal{B}_k = b_1 \vee \ldots \vee b_m$, we denote by $\boxed{\mathcal{B}_k} = \bigcap_{i=1}^{m} b_i$ (as sets of literals) the cube that consists of the literals that appear in all $b_1, \ldots, b_m$.

Fix a representation $\mathcal{B}_k = b_1 \vee \ldots \vee b_m$. Our main theorem is as follows:

Theorem 7.2. *Let $(Init, \delta, Bad)$ be a transition system. Then Λ-PDR$(Init, \delta, Bad, k)$ converges or fails in a frame whose index is bounded by*

$$\prod_{i=1}^{m} \left(\left| \mathcal{M}_{(Ref(\boxed{\mathcal{B}_k}), b_i)}(\delta) \right|_{\text{dnf}} + \left| \mathcal{M}_{b_i}(Init) \right|_{\text{dnf}} \right) + 1.$$

The reason for $\mathcal{M}_{b_i}(Init)$ will become clear in §7.1, and for $\boxed{\mathcal{B}_k}$ in §7.2. Often *Init* is a cube, in which case $\left| \mathcal{M}_{b_i}(Init) \right|_{\text{dnf}} = 1$.

Example 7.3. For an example where Thm. 7.2 yields a polynomial convergence bound, consider a counter over $\overline{x} = x_n, \ldots, x_0$ (similar in spirit to Fig. 1) with $Init = \overline{x} = 0$, $Bad = \overline{x} = 10\ldots0$, and δ that (i) skips every multiple of 2^r except $\overline{0} = 0\ldots0$, and (ii) transitions from any multiple of 2^r but $\overline{0}$ to any other multiple of 2^r (including the bad state). We call the set of states between consecutive multiples of $\overline{x} = 2^r, \overline{x} = 2^{r+1}$ a "segment".

Assume that $r = n - polylog(n)$ (the counter skips relatively few times). We now compute the bound resulting from the theorem. For every $k \geq 1$, $\mathcal{B}_k = \bigvee_{i=s}^{n} b_i$ where $b_i = (x_{r-1} = 0 \wedge \ldots \wedge x_0 = 0 \wedge x_i = 1)$ (all multiples of 2^r except $0\ldots0$). $\boxed{\mathcal{B}_k} = (x_{r-1} = 0 \wedge \ldots \wedge x_0 = 0)$ (all multiples of 2^r). We find a DNF representation for $\mathcal{M}_{(Ref(\boxed{\mathcal{B}_k}), b_i)}(\delta)$ using Lemma 4.5 similarly to §2.6: The number of segments is 2^{n-r}, and in each segment the abstraction of $n-r$ terms that introduce a new leading 1 subsumes the abstraction of increments that follow. This amounts to $(n-r)2^{n-r}$ terms. The number of disjuncts b_i is $n-r$, the number of terms in $\mathcal{M}_{b_i}(Init)$ is 1 because *Init* is itself a term, and overall Thm. 7.2 yields the bound $(n-r)^2 2^{n-r} = poly(n)$.

For an example where the theorem yields an exponential convergence bound, consider the same system but when $r = polylog(n)$. The above calculation still yields a bound of $(n-r)^2 2^{n-r}$ but in this case this is $\Omega(2^n)$. This exponential bound reflects true exponential behavior of the algorithm, because each post-image crosses to at most one new segment, and the abstraction never produces states in a segment beyond those represented in the current frame, mandating at least 2^{n-r} frames (see the extended version [Feldman et al. 2022] for details).

Example 7.4. The bound of Thm. 7.2 is not always tight. Consider the system from Example 4.12. Then $\boxed{\mathcal{B}_k} = true$, resulting in no abstraction of the pre-state vocabulary. The DNF size of $\mathcal{M}_{(true, b_i)}(\delta)$ is superpolynomial (see the extended version [Feldman et al. 2022]), leading to a superpolynomial bound on the number of frames. However, Λ-PDR with $k = 0$ converges in $\mathcal{F}_1$ (see Example 4.12).

Remark 7.1. *There are cases where it is possible to apply Thm. 7.2 on a restriction of δ to specific values to some of the variables, and this produces a better bound. Suppose that for a set of variables $\overline{x}$ and some $\overline{v}$ valuation thereof, $\overline{x} = \overline{v}$ is an inductive invariant for the system, and $\exists b \in \mathcal{B}_k.\ b[\overline{x}] = Ref(\overline{v})$. (In §2.6, actually $\mathcal{B}_k \implies \overline{x} = Ref(\overline{v})$.) Then applying Thm. 7.2 to $\delta\big|_{\overline{x} \leftarrow \overline{v}} = \delta[\overline{v}/\overline{x}]$, eliminating $\overline{x}$ by substituting $\overline{v}$ for it, also yields an upper bound on the number of iterations of Λ-PDR. The benefit is that $\left| \mathcal{M}_{(\ldots)}(\delta\big|_{\overline{x} \leftarrow \overline{v}}) \right|_{\text{dnf}}$ can be smaller than with the original δ. It is correct to apply the theorem to the restriction and deduce a bound for the original, because under the above premises, always $\mathcal{F}_i[\delta] = \mathcal{F}_i[\delta\big|_{\overline{x} \leftarrow \overline{v}}] \wedge \overline{x} = \overline{v}$, where $\mathcal{F}_i[\tau]$ is the ith frame of Λ-PDR w.r.t. transition relation τ.*[7]

To prove Thm. 7.2, the first step is to define an analog to the abstract transition system from Def. 6.3 that captures Alg. 3 in the general case. If $\mathcal{B}_k$ has a DNF form with m cubes, this can be done using a hypertransition system of width m (§7.1). We then proceed to bound its diameter (§7.2).

[7]This is because $\mathcal{F}_i \overset{\text{def}}{=} \mathcal{F}_i[\delta] \implies \overline{x} = \overline{v}$ by induction on i—since $\overline{x} = \overline{v}$ is an inductive invariant, this holds initially, as well as $\underline{\delta}(\mathcal{F}_i) \implies \overline{x} = \overline{v}$. Now $\mathcal{F}_{i+1} = \text{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i)) \implies \overline{x} = \overline{v}$ as well, because from the assumption there is $b \in \mathcal{B}_k$ s.t. $b[\overline{x}] = Ref(\overline{v})$, so $cube_b(\sigma) \implies \overline{x} = \overline{v}$ for every $\sigma \in \underline{\delta}(\mathcal{F}_i)$ because in σ, $\overline{x} = \overline{v}$, which are opposite in b and thus retained. Hence $\mathcal{M}_b(\underline{\delta}(\mathcal{F}_i)) = \bigvee_{\sigma \in \underline{\delta}(\mathcal{F}_i)} cube_b(\sigma) \implies \overline{x} = \overline{v}$, and thus also the conjunction $\text{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i)) \implies \overline{x} = \overline{v}$.

7.1 Abstract Hypertransition System

We consider hypertransition systems that are dual to the classical definition [e.g. Larsen and Liu 1990], in that the pre-state, instead of the post-state, of a hypertransition consists of a set of states.

Definition 7.5 (Hypertransition System). A *hypertransition system* (of width $m \in \mathbb{N}$) over States$[\Sigma]$ is a tuple $(\mathit{Init}, \delta, \mathit{Bad})$ where

- $\mathit{Init} \subseteq$ States$[\Sigma]$ is the set of initial states,
- $\mathit{Bad} \subseteq$ States$[\Sigma]$ is the set of bad states, and
- $\delta \subseteq$ States$[\Sigma]^m \times$ States$[\Sigma]$ is a hypertransition relation. As a formula, it is defined over m copies of Σ for the pre-states, $\Sigma_1, \ldots, \Sigma_m$, and a copy Σ' for the post-state.

An *execution* of the system is a tree in which the leaves are states from *Init*, the relationship between a node σ' and its children $\sigma_1, \ldots, \sigma_m$ is that $(\sigma_1, \ldots, \sigma_m, \sigma') \models \delta$. A state σ is *reachable in at most i steps* if there is an execution with root σ and height at most i. A state is *reachable* if it is reachable in at most i steps for some $i \in \mathbb{N}$. The *reachability diameter* of the system is the least i such that every reachable state is reachable in i steps.

A standard transition system is a hypertransition system with width $m = 1$.

Definition 7.6 (Abstract Hypertransition System). The abstract hypertransition system of a (standard) transition system $(\mathit{Init}, \delta, \mathit{Bad})$ w.r.t. $\mathcal{B}_k = b_1 \vee \ldots \vee b_m$ is defined over States$[\Sigma]$ by $\mathit{Init}^\sharp = \mathrm{MHull}_{\mathcal{B}_k}(\mathit{Init})$, $\mathit{Bad}^\sharp = \mathit{Bad}$, and $\delta^\sharp = \bigwedge_{i=1}^{m} (\mathcal{M}_{(\mathit{true}, b_i)}(\delta \vee \mathit{Init}'))[\Sigma_i, \Sigma']$. More explicitly (through Corollary 4.6),

$$(\sigma_1, \ldots, \sigma_m, \sigma') \in \delta^\sharp \iff \exists \sigma_1'', \ldots, \sigma_m''. \quad ((\sigma_1, \sigma_1'') \in \delta \vee \sigma_1'' \in \mathit{Init}) \wedge \sigma' \in \mathit{cube}_{b_1}(\sigma_1'') \quad \wedge$$
$$\ldots$$
$$((\sigma_m, \sigma_m'') \in \delta \vee \sigma_m'' \in \mathit{Init}) \wedge \sigma' \in \mathit{cube}_{b_m}(\sigma_m'').$$

In a hypertransition, the post-state σ' is reached from m "protector" states $\sigma_1'', \ldots, \sigma_m''$, each σ_i'' showing that σ' is in the monotone overapproximation w.r.t. one of the cubes b_i composing $\mathcal{B}_k$. Each protector state σ_i'' is reached via a step of the original transition system or is an initial state. The idea behind the connection between the reachable states of $\delta^\sharp$ and the Kleene iterations of Alg. 3 is that $\sigma' \in \mathrm{MHull}_{\mathcal{B}_k}(\{\sigma_1'', \ldots, \sigma_m''\})$, and if $\sigma_1, \ldots, \sigma_m \in \mathcal{F}_i^{\mathrm{ai}}$ then by Lemma 4.8 this implies that $\sigma' \in \mathrm{MHull}_{\mathcal{B}_k}(\delta(\mathcal{F}_i^{\mathrm{ai}}) \cup \mathit{Init})$, which, using the results of §5, is the next iterate $\mathcal{F}_{i+1}^{\mathrm{ai}}$. The key point is that the converse also holds—the monotone hull $\mathrm{MHull}_{\mathcal{B}_k}(\delta(\mathcal{F}_i^{\mathrm{ai}}) \cup \mathit{Init})$ "factors" to $\mathrm{MHull}_{\mathcal{B}_k}(\{\sigma_1'', \ldots, \sigma_m''\})$ on all m choices of protectors $\sigma_1'', \ldots, \sigma_m'' \in \delta(\mathcal{F}_i^{\mathrm{ai}}) \cup \mathit{Init}$ (this is reminiscent of Carathéodory's theorem in convex analysis). Unlike in Def. 6.3, in this definition the protector states also come directly from *Init*, not only from a transition of δ, and essentially this is the reason for $\mathcal{M}_{b_i}(\mathit{Init})$ in the bound of Thm. 7.2, unlike in Thm. 6.2. This is necessary here to "mix" the different protector states, which do not necessarily all originate from the same frame.

Lemma 7.7. *Let $R_i^\sharp$ be the set of states reachable in $(\mathit{Init}^\sharp, \delta^\sharp, \mathit{Bad}^\sharp)$ w.r.t. $\mathcal{B}_k$ (Def. 7.6) in at most i steps. Then $R_i^\sharp \equiv \mathcal{F}_i^{ai}$ where $\mathcal{F}_i^{ai}$ is the i'th iterate of Alg. 3 on $(\mathit{Init}, \delta, \mathit{Bad})$ in $\mathbb{M}[\mathcal{B}_k]$.*

Corollary 7.8. *Let $(\mathit{Init}^\sharp, \delta^\sharp, \mathit{Bad}^\sharp)$ be the abstract hypertransition system w.r.t. $\mathcal{B}_k$ (Def. 7.6). If $(\mathit{Init}^\sharp, \delta^\sharp, \mathit{Bad}^\sharp)$ is safe and its reachability diameter is s, then Λ-PDR$(\mathit{Init}, \delta, \mathit{Bad}, k)$ converges in frame at most $s + 1$. If $(\mathit{Init}^\sharp, \delta^\sharp, \mathit{Bad}^\sharp)$ reaches a bad state in s steps, then Λ-PDR$(\mathit{Init}, \delta, \mathit{Bad}, k)$ fails (line 8) in frame at most $s + 1$.*

7.2 (Hyper)Diameter Bounds via a Joint Abstract Cover

Lemma 7.9. *Let $(Init, \delta, Bad)$ be a transition system. Then the reachability diameter of $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ w.r.t. $\mathcal{B}_k$ (Def. 7.6) is at most $\prod_{i=1}^{m} \left(\left| \mathcal{M}_{(Ref(\boxed{\mathcal{B}_k}), b_i)}(\delta) \right|_{\text{dnf}} + \left| \mathcal{M}_{b_i}(Init) \right|_{\text{dnf}} \right)$.*

The proof is based on a diameter bound similar to the case of standard transition systems.

Lemma 7.10. *The reachability diameter of a hypertransition system $(Init, \delta, Bad)$ is bounded by $|\delta|_{\text{dnf}}$.*

As in Lemma 6.6, the proof of the theorem applies this bound to a variation on $\delta^\sharp$, which abstracts not only the post-state—that is, after performing a transition of the original system—but also the pre-states, before transitions to protector states. Similarly to Lemma 6.6, the result is a different hypertransition system, but its diameter is the same as of the abstract hypertransition system per Def. 7.6. The reason to use $\boxed{\mathcal{B}_k}$ is to reduce the diameter bound without reducing the diameter. To do so, if $(\sigma_1, \ldots, \sigma_m, \sigma') \in \delta^\sharp$, then for each σ_i, we want to generalize it as much as possible to also consider σ_i' that can arrive at σ_i using monotonization w.r.t. $\text{MHull}_{\mathcal{B}_k}(\cdot)$, as already used in $\delta^\sharp$ in the *post-state*. This ensures that reachability is not extended, as the additional abstraction in the pre-state could be mimicked by an abstraction in the post-state of the previous (abstract) step. To ensure that σ_i' arrives at σ_i using monotonization w.r.t. $\text{MHull}_{\mathcal{B}_k}(\cdot)$, we need to ensure that for every b_j, $\sigma_i \in \mathcal{M}_{b_j}(\sigma_i')$. This is achieved when $\sigma_i \in \mathcal{M}_{\boxed{\mathcal{B}_k}}(\sigma_i')$, i.e., $\sigma_i' \in \mathcal{M}_{Ref(\boxed{\mathcal{B}_k})}(\sigma_i)$.

Combining Corollary 7.8 and Lemma 7.9 yields a proof of Thm. 7.2, similarly to Thm. 6.2.

8 FORWARD REACHABILITY IN Λ-PDR AND OTHERS

This section highlights the importance of the successive overapproximation embodied in the Kleene iterations of Λ-PDR by contrasting Λ-PDR with the treatment of forward reachability in other invariant inference algorithms.

Exact forward reachability. Exact forward reachability iterates $R_0 = Init, R_{i+1} = \underline{\delta}(R_i)$, so that R_i is the set of states reachable in at most i steps (without any overapproximation). We have shown that in some cases Λ-PDR can converge in a significantly lower number of iterations than exact forward reachability, stated formally in the following lemma.

Lemma 8.1. *There exists a family of transition systems $(Init, \delta, Bad)$ over Σ with $|\Sigma| = n$ and $k = O(1)$ such that Λ-PDR$(Init, \delta, Bad, k)$ converges in poly(n, k) iterations, whereas exact forward reachability converges in $\Omega(2^n)$ iterations.*

Proof. See e.g. §2.5 and Example 4.11. □

This gap reflects a gap between the diameter of the original system $(Init, \delta)$ and the diameter of the abstract system $(Init^\sharp, \delta^\sharp)$ (Lemma 7.9).

Dual interpolation. The essence of *interpolation-based inference* (ITP) [McMillan 2003] is generalizing from proofs of *bounded* unreachability. We consider the time-dual [e.g. Feldman et al. 2020, Appendix A] of this approach, generalizing from bounded unreachabilty *from* the initial states, rather than unreachability *to* the bad states, in line with our focus here on the treatment of forward reachability. Specifically, Alg. 4 is based on (the time-dual of) a model-based ITP algorithm [Bjørner et al. 2013; Chockler et al. 2012] whose generalization procedure was inspired by PDR.

The algorithm is parametrized by a forward-exploration bound s. It refines a candidate φ starting from the candidate that excludes just the bad states (line 2). In each iteration, the algorithm samples a pre-state σ_b of a counterexample to the induction of φ, a state in φ that in one step reaches states outside φ (line 4). Instead of excluding just the counterexample—similarly to PDR—

the algorithm seeks a minimal clause c over the literals that are falsified in σ_b that does not exclude a state from $\mathcal{R}_s$, the set of states that the system can reach in s steps (line 7), and conjoins c to the candidate (line 8).

The complexity of this algorithm was recently studied by Feldman et al. [2021], who showed that the forward-exploration bound s is sufficient to discover an inductive invariant when s steps reach the entire *inner boundary* of I,

Algorithm 4 Dual Model-Based ITP, based on [Bjørner et al. 2013; Chockler et al. 2012]

```
1: procedure Dual-Model-Based-ITP(Init,δ,Bad,s)
2:     φ ← ¬Bad
3:     while φ not inductive do
4:         take σ_b ∈ φ s.t. δ(σ_b) ⊈ φ
5:         if σ_b ∈ R_s then
6:             restart with larger s
7:         take minimal c ⊆ ¬σ_b s.t. R_s ⟹ c
8:         φ ← φ ∧ c
9:     return φ
```

$$\partial^+(I) \stackrel{\text{def}}{=} \{\sigma^+ \mid \exists \sigma^-.\ \sigma^+ \models I,\ \sigma^- \models \neg I,\ \text{Hamming-Distance}(\sigma^+, \sigma^-) = 1\}.$$

Theorem 8.2 (Feldman et al. [2021]). *Let I be an inductive invariant for $(Init, \delta, Bad)$, and $\mathcal{R}_s$ the set of states reachable in at most s steps in $(Init, \delta, Bad)$. If $\partial^+(I) \subseteq \mathcal{R}_s$, then a forward bound of s suffices for Dual-Model-Based-ITP$(Init, \delta, Bad, s)$ to successfully find an inductive invariant.*

In the example of Fig. 1 (from §2), this does not hold for the invariant in Equation (1) unless $s = \Omega(2^n)$ (for example, $\overline{x} = 110\ldots0, \overline{y} = 0\ldots0, z = 0 \in \partial^+(I)$ but reachable only in $\Omega(2^n)$ steps).

In contrast, we prove that for Λ-PDR, it is enough that $\partial^+(I)$ is s-reachable in the *abstract* (hyper)system, which interleaves concrete steps and abstraction (see Def. 7.6), and thus can reach $\partial^+(I)$ in fewer steps, which would result in convergence of Λ-PDR with a smaller number of frames:

Theorem 8.3. *Let $I \in \text{MSpan}(\mathcal{B}_k)$ be an inductive invariant for $(Init, \delta, Bad)$, and $\mathcal{R}_s^\sharp$ the set of states reachable in at most s steps in $(Init^\sharp, \delta^\sharp, Bad^\sharp)$ (Def. 7.6). If $\partial^+(I) \subseteq \mathcal{R}_s^\sharp$, then $s+1$ frames suffice for Λ-PDR$(Init, \delta, Bad, k)$ to successfully find an inductive invariant.*

The two results can be proved similarly, using tools from the monotone theory. In both cases, the argument is that when s is large enough, the monotone hull of the current candidate must include the entire I. In Alg. 4, the argument is that always $I \subseteq \text{MHull}_{C_i}(\mathcal{R}_s) \subseteq \varphi$, where C_i is the set of counterexamples σ_b that the algorithm has encountered so far. In Alg. 2, the argument is that $I \subseteq \text{MHull}_{\mathcal{B}_k}(\mathcal{R}_s^\sharp)$. Both rely on the following fact about the monotone hull of a boundary of a set:

Lemma 8.4. *Let I, S, B be sets of states s.t. $\partial^+(I) \subseteq S$ and $B \cap I = \emptyset$. Then $I \subseteq \text{MHull}_B(S)$.*

Proof. Let $\sigma \in I$. For every $b \in B$, assume for the sake of contradiction that σ is not in $\mathcal{M}_b(S)$. By Thm. 4.9, there is some cube e such that $b \models e$ and $\sigma \models e$ but $S \implies \neg e$. In particular, $\partial^+(I) \implies \neg e$. Consider some shortest path between σ, b in the Hamming cube. Because $\sigma \models I, b \not\models I$, there is some crossing point $\sigma^+ \in \partial^+(I)$ on that path. This state σ^+ agrees with σ, b on the literals on which they agree, which include all the literals in e, since this is a cube (a conjunction of literals). Hence also $e \models \sigma^+$, but this is a contradiction to $\partial^+(I) \implies \neg e$. □

Proofs of Thm. 8.2 and Thm. 8.3 are derived from Lemma 8.4 in the extended version [Feldman et al. 2022]. In essence, these different criteria for when the forward-exploration of the algorithm is sufficient reflect the difference in how the algorithms generalize: per counterexample, both find a minimal clause that does not exclude states from some form of forward reachability, but in Λ-PDR this is an abstraction of forward reachability, whereas Alg. 4 uses exact forward reachability.

This difference also manifests in different outcomes of Alg. 2 and Alg. 4 on the running example of Fig. 1. For every $s < 2^n$ there is an execution of Alg. 4 that fails (line 5) because it includes reachable states as counterexamples to exclude (for example, the first counterexample in the execution of Alg. 4 is $\sigma_b = (\overline{x} = 10\ldots00, \overline{y} = 11\ldots10, z = 1)$, which can be generalized to $c = (x_n = 0)$ that

inadvertently excludes also reachable states such as $\overline{x} = 10\ldots01, \overline{y} = 00\ldots00, z = 0$), although $s = O(1)$ suffices for Λ-PDR (Example 4.11).

Finally, we remark that Alg. 4 does use a form of successive overapproximation. By repeatedly generating counterexamples to induction (line 4), it in a sense uses reverse frames that overapproximate backward reachability. While both Alg. 4 and Alg. 2 learn lemmas by minimizing a term w.r.t. a forward-reachability analysis in order to block a counterexample from a backward-reachability analysis, Alg. 2 employs successive overapproximation is in the former analysis, and Alg. 4 in the latter. As we have seen, this successive overapproximation in counterexample generation is not sufficient for Alg. 4 to successfully infer an invariant for the example of Fig. 1. However, it does alleviate the requirement that $I \in \mathcal{B}_k$, which is necessary in Thm. 8.3 but not in Thm. 8.2.[8]

9 BETWEEN Λ-PDR AND PDR: BEST ABSTRACTION AND EVEN BETTER

In each frame, Λ-PDR includes all possible generalizations, which we have shown to amount in $\mathcal{F}_{i+1}$ to the the best abstraction of $\underline{\delta}(\mathcal{F}_i)$ in the abstract domain $\mathbb{M}[\mathcal{B}_k]$ (Lemma 5.4). Its frames are thus the strongest (contain fewest states) that satisfy all the properties of frames listed in §2.1—the standard ones as well as the monotone span of backward reachable states:

Lemma 9.1. *The frames $\mathcal{F}_0, \mathcal{F}_1, \ldots$ of Λ-PDR are the least (w.r.t. $\Longrightarrow$) s.t. for every i, (1) $Init \Longrightarrow \mathcal{F}_0$, (2) $\mathcal{F}_i \Longrightarrow \mathcal{F}_{i+1}$, (3) $\delta(\mathcal{F}_i) \Longrightarrow \mathcal{F}_{i+1}$, and (5) $\mathcal{F}_i \in \mathrm{MSpan}(\mathcal{B}_k)$.*

In contrast to Λ-PDR, standard PDR "samples" counterexamples and generalizations, and it does not produce in $\mathcal{F}^{\mathrm{pdr}}_{i+1}$ the least abstraction of $\underline{\delta}(\mathcal{F}^{\mathrm{pdr}}_i)$. Its frames are nevertheless characterized as abstractions (not necessarily the least abstraction) in the same domain:

Lemma 9.2. *At any point during the execution of PDR($Init, \delta, Bad$) (Alg. 1) when it has at most N frames, $\mathcal{F}^{pdr}_i \in \mathrm{MSpan}(\mathcal{B}_N)$ for every $1 \le i \le N$.*

In particular, this shows that PDR overapproximates the frames that Λ-PDR generates:

Corollary 9.3. *At any point during the execution of PDR($Init, \delta, Bad$) (Alg. 1) when it has at most N frames, its i'th frame, $\mathcal{F}^{pdr}_i$, satisfies $\mathcal{F}_i \Longrightarrow \mathcal{F}^{pdr}_i$, where $\mathcal{F}_i$ is the i'th frame of Λ-PDR($Init, \delta, Bad, N$) (Alg. 2).*

In other words, PDR's frame also constitute some sort of search in the abstract domain $\mathbb{M}[\mathcal{B}_N]$ (though in a complex manner, refining previous frame etc.), and its frames always generate at least as much overapproximation as Λ-PDR. Hence, our results that show significant overapproximation in Λ-PDR translate to PDR as well.

Still, the difference between the algorithms is significant—PDR's frames don't employ the best abstraction in this domain. How does this benefit PDR? We show two ways. First, computing all generalizations may be inefficient. Second, it may not be desirable—it could lead to too precise abstraction and slow convergence.

Inefficient frame size. Consider a system over n variables $x_1, \ldots, x_n$, with $Init = x_1 = \ldots = x_n = 0$, $Bad = x_1 = \ldots = x_n = 1$, and δ that non-deterministically chooses some $i \neq j$ with $x_i = x_j = 0$ and sets $x_i \leftarrow 1$.

We start with the analysis of Λ-PDR. In this example, $\mathcal{B}_k = 1\ldots1$ (for every k). We argue that $\mathcal{F}_i$ is exactly the set R_i of states reachable in at most i steps, which is the set of states with

[8] The original, non time-dual version of the algorithm, has frames going forward, such as in PDR, but the roles of backward- and forward-reachability in generalization are reversed. This algorithm "overshoots" on the example of Fig. 1 unless $s = \Omega(2^n)$, but we focus here on overapproximations that are too tight (rather than too loose), the direction in which Λ-PDR is informative of PDR.

at most i bits 1, denoted $\{\overline{x} \mid \#1(\overline{x}) \leq i\}$. This can be seen by induction: initially, this holds for $\mathcal{F}_0 = \mathit{Init} = \{0 \dots 0\}$. In each step $\delta(\mathcal{F}_i) = R_i \cup \{\overline{x} \mid \#1(\overline{x}) = i+1\}$. Then $\mathcal{F}_{i+1} = \mathrm{MHull}_{\mathcal{B}_k}(\underline{\delta}(\mathcal{F}_i)) = \mathcal{M}_{1\dots1}(\underline{\delta}(\mathcal{F}_i)) = R_i \cup \mathcal{M}_{1\dots1}(\{\overline{x} \mid \#1(\overline{x}) = i+1\}) = R_{i+1}$, because $\mathcal{M}_{1\dots1}(\{\overline{x} \mid \#1(\overline{x}) = i+1\})$ adds states that are obtained from a state with $\#1(\overline{x}) = i+1$ by flipping 1's to 0's, resulting in states with smaller values of $\#1(\overline{x})$ that are already included in R_i.

Unfortunately, the set $R_{\lfloor n/2 \rfloor + 1}$ is not expressible in polynomial-size CNF nor DNF.[9] This means that some of Λ-PDR's frames need an exponential number of clauses, and so construct an exponential number of generalizations of the bad state. Even an alternative DNF computation (based on Lemma 4.5) would not fare better.

In contrast, $\mathcal{F}_i^{\mathrm{pdr}}$ consists of a single clause blocking the bad state, which is short.

Slow convergence. Consider a counter over $\overline{x} = x_n, x_{n-1}, \dots, x_0$ with $\mathit{Init} = (\overline{x} = 0 \dots 0)$, $\mathit{Bad} = (\overline{x} = 1 \dots 1)$, and δ that increments the counter except for when $\overline{x} = 1 \dots 10$ which skips the bad state and wraps-around to $0 \dots 0$.

We start with the analysis of Λ-PDR. Similar to the previous example, $\mathcal{B}_k = 1 \dots 1$ (for every k) and $\mathcal{F}_i = R_i$, except that R_i is now the set of states $\overline{x} \leq i$, because $\delta(\mathcal{F}_i)$ always adds the state $\overline{x} = i+1$, and its $1 \dots 1$-monotonization adds only states with smaller values of $\overline{x}$ which are already included in R_i (the derivation is similar to the previous example).

Therefore, the frames $\mathcal{F}_i = \{\overline{x} : \overline{x} \leq i\}$ do not converge until $i = 2^n - 1$, which means that Λ-PDR converges after an exponential number of frames.

In contrast, in this example, PDR always converges in a *linear* number of frames. The proof uses the fact from Lemma 9.2 that the frames of PDR are $(1 \dots 1)$-monotone, and that $\mathcal{F}_i^{\mathrm{pdr}}$ is always exactly one clause, because it blocks the single backward reachable state using a single lemma. Since $\mathcal{F}_i^{\mathrm{pdr}} \implies \mathcal{F}_{i+1}^{\mathrm{pdr}}$, and $\mathcal{F}_i^{\mathrm{pdr}}$ is 1...1-monotone, the clause that is $\mathcal{F}_{i+1}^{\mathrm{pdr}}$ must be a syntactic subset of the clause that is $\mathcal{F}_i^{\mathrm{pdr}}$ [Quine 1954]. Until they converge, the difference between two successive frames must be that some literals are omitted from the clause, which can happen at most n times.

10 RELATED WORK

PDR as abstract interpretation. This work is not the first to study the relation between PDR and abstract interpretation. Rinetzky and Shoham [2016] prove that the reachable configurations of PDR are in simulation with the reachable states of a non-standard backward trace semantics. Their work studies standard PDR as non-standard abstract interpretation, whereas we study non-standard PDR as standard abstract interpretation (in a new domain); our domain abstracts the simpler collecting states semantics with standard forward iterations. Our work emphasizes the overapproxmation inherent in the abstraction, where, in particular, the abstraction forces overapproximation in the sequence of frames, whereas Rinetzky and Shoham's property-guided Cartesian trace semantics domain is precise enough to express any sequence of frames that satisfy properties 1–4 from §2.1. In contrast, adding property 5 characterizes Λ-PDR as Kleene iterations in our $\mathbb{M}[\mathcal{B}_k]$ domain.

Abstract transition systems. Dams et al. [1997] construct, from a transition system and a Galois connection, abstract transition systems that preserve safety and other temporal properties. These are defined over a state space of abstract elements (e.g., formulas in the case of a logical domain), forming abstract edges between abstract elements through $\exists\exists$ or $\forall\exists$ relations of original transitions between the concretizations. It is important for our diameter bounds from the DNF representation of the abstract (hyper)transition system that it is defined over the original state space (Defs. 6.3 and 7.6), which is possible due to the special structure of $\mathbb{M}[B]$ (see Lemmas 6.4 and 7.7). In that respect our abstract transition systems are closer to monotonic abstraction in well-structured transition systems

[9] It is the majority function, which is not in AC^0 [Håstad 1986], a complexity class that includes poly-size CNF and DNF.

by Abdulla et al. [2009], the abstract transition systems for universally-quantified uninterpreted domains by Padon et al. [2016], and the surjective abstraction games of Fecher and Huth [2007].

Diameter bounds. Diameter bounds have been studied in the context of completeness thresholds for bounded model checking [Biere et al. 1999a; Kroening and Strichman 2003]. The recurrence diameter [Biere et al. 1999b; Kroening and Strichman 2003], the longest loop-free path, was studied as a more easily-computable upper bound on the diameter. In our setting, this measure cannot be reduced by the abstraction, which only adds transitions. There are also works that encode the completeness threshold assumption as another verification condition [see D'Silva et al. 2008, §IV.D]. Another line of work computes diameter bounds by a composition of diameter bounds of subsystems formed by separating dependencies between variables in the system's actions [Abdulaziz et al. 2018; Baumgartner et al. 2002; Rintanen and Gretton 2013]. Existing works have considered guarded-update actions, in which variables are either modified to a constant value or remain unchanged; this is not directly applicable to the actions that arise in our abstract transition systems, where monotonization in effect "havocs" variables. Havocked variables are different because in a transition, they *can* change, but not *necessarily*; weaker notions of dependence to capture this may be interesting in future work. We are not aware of a previous application of the $|\delta|_{\text{dnf}}$ diameter bound (Lemma 6.7). This bound is never worsened by monotonization, as $|\mathcal{M}_{\dots}(\delta)|_{\text{dnf}} \leq |\delta|_{\text{dnf}}$ [Bshouty 1995, and a corollary of Lemma 4.5], and can be exponentially smaller, as e.g. in §2.6. The diameter bounds by Konnov et al. [2014, 2017] share with our work the motivation of analyzing the diameter of abstractions of the original system. They rely on the special structure of counter abstractions of fault-tolerant distributed systems to apply movers [Lipton 1975] and acceleration.

Complexity of invariant inference algorithms. Houdini [Flanagan and Leino 2001] infers conjunctive invariants in a linear number of SAT calls, and Dual Houdini likewise for disjunctive invariants [Lahiri and Qadeer 2009]. Feldman et al. [2021] analyze the complexity of an interpolation-based inference algorithm based on the fence condition, which we compare with our results in §8. The work by Seufert and Scholl [2017] includes a complexity analysis of all executions of PDR on the case of two synchronized n-bit counters, where PDR requires an exponential number of SAT calls (this also follows from the fact that the only CNF invariant is exponentially-large) but an enhanced time-dual version of it converges in one frame. Convergence in one frame is also proved for maximal transitions systems with monotone invariants [Feldman et al. 2020]. In §9 we go beyond this with an analysis of standard PDR on a simple example where convergence requires multiple frames. Our analysis of Λ-PDR centered on the number of frames, not the complexity of constructing them, which is an interesting direction for future work. Although, in the spirit of §6, we can bound $|\mathcal{F}_i|_{\text{dnf}} \leq \left|\delta^\sharp\right|_{\text{dnf}} = |\mathcal{M}_b(\delta)|_{\text{dnf}}$ (when $b = \mathcal{B}_k$ is a cube), the original Λ-algorithm's complexity analysis [Bshouty 1995] for computing $\mathcal{M}_b(\varphi)$ depends on $|\varphi|_{\text{dnf}}$, not $|\mathcal{M}_b(\varphi)|_{\text{dnf}}$, which in our setting is the difference between the concrete and the significantly reduced abstract diameter.

The monotone theory in invariant inference. The monotone theory [Bshouty 1995] has been used for other purposes in invariant inference. Jung et al. [2015] use Bshouty's CDNF learning algorithm to infer predicate abstraction invariants, employing over- and under-approximations to resolve membership queries, sometimes relying on random guesses. Feldman et al. [2021] use Bshouty's Λ-algorithm for provably-efficient inference of an invariant whose s-reachable (cf. Thm. 8.2) and belongs to $\text{MSpan}(B)$ when B is known a-priori. Chen et al. [2010] use the CDNF algorithm for automatic generation of contextual assumptions in assume-guarantee.

11 CONCLUSION

This work has distilled a previously unknown principle of property-directed reachability. Through Λ-PDR and its analysis based on the monotone theory from exact learning, we have shown that PDR overapproximates an abstract interpretation process in a new logical abstract domain. We have further shown how this abstraction achieves a significantly more effective forward reachability exploration than approaches that use exact post-image computations or bounded unrollings, and how this can partially be explained through the difference between diameter bounds between the original system and its abstraction.

In future work, it will be interesting to understand the mechanisms by which PDR deviates from naive backward reachability, avoiding the pitfall in the other direction, of overapproximating too much. We hope that this will eventually lead to efficient complexity results for PDR itself. It will also be interesting to study variants of PDR that target infinite-state using richer logics beyond propositional logic. Our observation that there is inherent abstraction in PDR due to states it *cannot* exclude from a frame may also be relevant in such settings. This could also involve extensions of the monotone theory to other logics, which to our knowledge have not been attempted.

ACKNOWLEDGMENTS

We thank our shepherd and the anonymous reviewers for comments which improved the paper. We thank Mohammad Abdulaziz, Aman Goel, Alexander Ivrii, Noam Parzanchevski, Hila Peleg, and Noam Rinetzky for insightful discussions and comments. The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement No [759102-SVIS]). This research was partially supported by the United States-Israel Binational Science Foundation (BSF) grant No. 2016260, and the Israeli Science Foundation (ISF) grant No. 1810/18.

REFERENCES

Mohammad Abdulaziz, Michael Norrish, and Charles Gretton. 2018. Formally Verified Algorithms for Upper-Bounding State Space Diameters. *J. Autom. Reason.* 61, 1-4 (2018), 485–520. https://doi.org/10.1007/s10817-018-9450-z

Parosh Aziz Abdulla, Giorgio Delzanno, Noomene Ben Henda, and Ahmed Rezine. 2009. Monotonic Abstraction: on Efficient Verification of Parameterized Systems. *Int. J. Found. Comput. Sci.* 20, 5 (2009), 779–801. https://doi.org/10.1142/S0129054109006887

Aws Albarghouthi, Yi Li, Arie Gurfinkel, and Marsha Chechik. 2012. Ufo: A Framework for Abstraction- and Interpolation-Based Software Verification. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings (Lecture Notes in Computer Science, Vol. 7358)*, P. Madhusudan and Sanjit A. Seshia (Eds.). Springer, 672–678. https://doi.org/10.1007/978-3-642-31424-7_48

Jason Baumgartner, Andreas Kuehlmann, and Jacob A. Abraham. 2002. Property Checking via Structural Analysis. In *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2404)*, Ed Brinksma and Kim Guldstrand Larsen (Eds.). Springer, 151–165. https://doi.org/10.1007/3-540-45657-0_12

Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Masahiro Fujita, and Yunshan Zhu. 1999b. Symbolic Model Checking Using SAT Procedures instead of BDDs. In *Proceedings of the 36th Conference on Design Automation, New Orleans, LA, USA, June 21-25, 1999*, Mary Jane Irwin (Ed.). ACM Press, 317–320. https://doi.org/10.1145/309847.309942

Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. 1999a. Symbolic Model Checking without BDDs. In *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings*. 193–207. https://doi.org/10.1007/3-540-49059-0_14

Nikolaj Bjørner, Arie Gurfinkel, Konstantin Korovin, and Ori Lahav. 2013. Instantiations, Zippers and EPR Interpolation. In *LPAR 2013, 19th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, December 12-17, 2013, Stellenbosch, South Africa, Short papers proceedings*. 35–41. https://easychair.org/publications/paper/XtN

Aaron R. Bradley. 2011. SAT-Based Model Checking without Unrolling. In *Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings*. 70–87. https://doi.org/10.1007/978-3-642-18275-4_7

Nader H. Bshouty. 1995. Exact Learning Boolean Function via the Monotone Theory. *Inf. Comput.* 123, 1 (1995), 146–153. https://doi.org/10.1006/inco.1995.1164

Yu-Fang Chen, Edmund M. Clarke, Azadeh Farzan, Ming-Hsien Tsai, Yih-Kuen Tsay, and Bow-Yaw Wang. 2010. Automated Assume-Guarantee Reasoning through Implicit Learning. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*. 511–526. https://doi.org/10.1007/978-3-642-14295-6_44

Hana Chockler, Alexander Ivrii, and Arie Matsliah. 2012. Computing Interpolants without Proofs. In *Hardware and Software: Verification and Testing - 8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6-8, 2012. Revised Selected Papers*. 72–85. https://doi.org/10.1007/978-3-642-39611-3_12

Edmund M. Clarke and E. Allen Emerson. 1981. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981 (Lecture Notes in Computer Science, Vol. 131)*, Dexter Kozen (Ed.). Springer, 52–71. https://doi.org/10.1007/BFb0025774

Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*. 238–252. https://doi.org/10.1145/512950.512973

P. Cousot and R. Cousot. 1979. Systematic Design of Program Analysis Frameworks. In *Symp. on Princ. of Prog. Lang.* ACM Press, New York, NY, 269–282.

Dennis Dams, Rob Gerth, and Orna Grumberg. 1997. Abstract Interpretation of Reactive Systems. *ACM Trans. Program. Lang. Syst.* 19, 2 (1997), 253–291. https://doi.org/10.1145/244795.244800

Vijay D'Silva, Daniel Kroening, and Georg Weissenbacher. 2008. A Survey of Automated Techniques for Formal Software Verification. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 27, 7 (2008), 1165–1178. https://doi.org/10.1109/TCAD.2008.923410

Niklas Eén, Alan Mishchenko, and Robert K. Brayton. 2011. Efficient implementation of property directed reachability. In *International Conference on Formal Methods in Computer-Aided Design, FMCAD '11, Austin, TX, USA, October 30 - November 02, 2011*. 125–134. http://dl.acm.org/citation.cfm?id=2157675

P. Ezudheen, Daniel Neider, Deepak D'Souza, Pranav Garg, and P. Madhusudan. 2018. Horn-ICE learning for synthesizing invariants and contracts. *PACMPL* 2, OOPSLA (2018), 131:1–131:25.

Harald Fecher and Michael Huth. 2007. More Precise Partition Abstractions. In *Verification, Model Checking, and Abstract Interpretation, 8th International Conference, VMCAI 2007, Nice, France, January 14-16, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4349)*, Byron Cook and Andreas Podelski (Eds.). Springer, 167–181. https://doi.org/10.1007/978-3-540-69738-1_12

Yotam M. Y. Feldman, Neil Immerman, Mooly Sagiv, and Sharon Shoham. 2020. Complexity and information in invariant inference. *Proc. ACM Program. Lang.* 4, POPL (2020), 5:1–5:29. https://doi.org/10.1145/3371073

Yotam M. Y. Feldman, Mooly Sagiv, Sharon Shoham, and James R. Wilcox. 2021. Learning the boundary of inductive invariants. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–30. https://doi.org/10.1145/3434296

Yotam M. Y. Feldman, Mooly Sagiv, Sharon Shoham, and James R. Wilcox. 2022. Property-Directed Reachability as Abstract Interpretation in the Monotone Theory. *CoRR* (2022). https://arxiv.org/pdf/2111.00324.pdf

Cormac Flanagan and K. Rustan M. Leino. 2001. Houdini, an Annotation Assistant for ESC/Java. In *FME 2001: Formal Methods for Increasing Software Productivity, International Symposium of Formal Methods Europe, Berlin, Germany, March 12-16, 2001, Proceedings*. 500–517.

Cormac Flanagan and Shaz Qadeer. 2002. Predicate abstraction for software verification. In *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*. 191–202. https://doi.org/10.1145/503272.503291

Pranav Garg, Christof Löding, P Madhusudan, and Daniel Neider. 2014. ICE: A robust framework for learning invariants. In *Computer Aided Verification*. Springer, 69–87.

Pranav Garg, Daniel Neider, P. Madhusudan, and Dan Roth. 2016. Learning invariants using decision trees and implication counterexamples. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*. 499–512. https://doi.org/10.1145/2837614.2837664

Susanne Graf and Hassen Saïdi. 1997. Construction of Abstract State Graphs with PVS. In *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*. 72–83. https://doi.org/10.1007/3-540-63166-6_10

Sumit Gulwani, Bill McCloskey, and Ashish Tiwari. 2008. Lifting abstract interpreters to quantified logical domains. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, George C. Necula and Philip Wadler (Eds.). ACM, 235–246. https://doi.org/10.1145/1328438.1328468

Arie Gurfinkel and Alexander Ivrii. 2015. Pushing to the Top. In *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015*. 65–72.

Arie Gurfinkel and Alexander Ivrii. 2017. K-induction without unrolling. In *2017 Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, October 2-6, 2017*, Daryl Stewart and Georg Weissenbacher (Eds.). IEEE, 148–155. https://doi.org/10.23919/FMCAD.2017.8102253

Arie Gurfinkel, Sharon Shoham, and Yuri Meshman. 2016. SMT-based verification of parameterized systems. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. 338–348. https://doi.org/10.1145/2950290.2950330

Johan Håstad. 1986. Almost Optimal Lower Bounds for Small Depth Circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, Juris Hartmanis (Ed.). ACM, 6–20. https://doi.org/10.1145/12130.12132

Edith Hemaspaandra, Lane A. Hemaspaandra, Till Tantau, and Osamu Watanabe. 2010. On the complexity of kings. *Theor. Comput. Sci.* 411, 4-5 (2010), 783–798. https://doi.org/10.1016/j.tcs.2009.10.015

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. 2010. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*. 215–224. https://doi.org/10.1145/1806799.1806833

Susmit Jha and Sanjit A. Seshia. 2017. A theory of formal synthesis via inductive learning. *Acta Inf.* 54, 7 (2017), 693–726. https://doi.org/10.1007/s00236-017-0294-5

Yungbum Jung, Soonho Kong, Cristina David, Bow-Yaw Wang, and Kwangkeun Yi. 2015. Automatically inferring loop invariants via algorithmic learning. *Math. Struct. Comput. Sci.* 25, 4 (2015), 892–915. https://doi.org/10.1017/S0960129513000078

Jason R. Koenig, Oded Padon, Neil Immerman, and Alex Aiken. 2020. First-order quantified separators. In *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, Alastair F. Donaldson and Emina Torlak (Eds.). ACM, 703–717. https://doi.org/10.1145/3385412.3386018

Igor Konnov, Helmut Veith, and Josef Widder. 2014. On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8704)*, Paolo Baldan and Daniele Gorla (Eds.). Springer, 125–140. https://doi.org/10.1007/978-3-662-44584-6_10

Igor V. Konnov, Marijana Lazic, Helmut Veith, and Josef Widder. 2017. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, Giuseppe Castagna and Andrew D. Gordon (Eds.). ACM, 719–734. https://doi.org/10.1145/3009837.3009860

Daniel Kroening and Ofer Strichman. 2003. Efficient Computation of Recurrence Diameters. In *Verification, Model Checking, and Abstract Interpretation, 4th International Conference, VMCAI 2003, New York, NY, USA, January 9-11, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2575)*, Lenore D. Zuck, Paul C. Attie, Agostino Cortesi, and Supratik Mukhopadhyay (Eds.). Springer, 298–309. https://doi.org/10.1007/3-540-36384-X_24

Shuvendu K. Lahiri and Shaz Qadeer. 2009. Complexity and Algorithms for Monomial and Clausal Predicate Abstraction. In *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*. 214–229.

Kim Guldstrand Larsen and Xinxin Liu. 1990. Equation Solving Using Modal Transition Systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*. IEEE Computer Society, 108–117. https://doi.org/10.1109/LICS.1990.113738

Richard J. Lipton. 1975. Reduction: A Method of Proving Properties of Parallel Programs. *Commun. ACM* 18, 12 (1975), 717–721. https://doi.org/10.1145/361227.361234

Kenneth L. McMillan. 2003. Interpolation and SAT-Based Model Checking. In *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*. 1–13.

Daniel Neider, P. Madhusudan, Shambwaditya Saha, Pranav Garg, and Daejun Park. 2020. A Learning-Based Approach to Synthesizing Invariants for Incomplete Verification Engines. *J. Autom. Reason.* 64, 7 (2020), 1523–1552. https://doi.org/10.1007/s10817-020-09570-z

Oded Padon, Neil Immerman, Sharon Shoham, Aleksandr Karbyshev, and Mooly Sagiv. 2016. Decidability of inferring inductive invariants. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*. 217–231. https://doi.org/10.1145/2837614.2837640

Jean-Pierre Queille and Joseph Sifakis. 1982. Specification and verification of concurrent systems in CESAR. In *International Symposium on Programming, 5th Colloquium, Torino, Italy, April 6-8, 1982, Proceedings (Lecture Notes in Computer Science, Vol. 137)*, Mariangiola Dezani-Ciancaglini and Ugo Montanari (Eds.). Springer, 337–351. https://doi.org/10.1007/3-540-11494-7_22

WV Quine. 1954. Two theorems about truth-functions. *Boletín de la Sociedad Matemática Mexicana* 10, 1–2 (1954), 64–70.

Noam Rinetzky and Sharon Shoham. 2016. Property Directed Abstract Interpretation. In *Verification, Model Checking, and Abstract Interpretation - 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016. Proceedings (Lecture Notes in Computer Science, Vol. 9583)*, Barbara Jobstmann and K. Rustan M. Leino (Eds.). Springer,

104–123. https://doi.org/10.1007/978-3-662-49122-5_5

Jussi Rintanen and Charles Orgill Gretton. 2013. Computing Upper Bounds on Lengths of Transition Sequences. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, Francesca Rossi (Ed.). IJCAI/AAAI, 2365–2372. http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6992

Xavier Rival and Kwangkeun Yi. 2020. *Introduction to Static Analysis: An Abstract Interpretation Perspective.* MIT Press.

Marcus Schaefer and Christopher Umans. 2002. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT news* 33, 3 (2002), 32–49.

Tobias Seufert and Christoph Scholl. 2017. Sequential Verification Using Reverse PDR. In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, MBMV 2017, Bremen, Germany, February 8-9, 2017*, Daniel Große and Rolf Drechsler (Eds.). Shaker Verlag, 79–90.

Rahul Sharma and Alex Aiken. 2016. From invariant checking to invariant inference using randomized search. *Formal Methods in System Design* 48, 3 (2016), 235–256. https://doi.org/10.1007/s10703-016-0248-5

Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken, Percy Liang, and Aditya V. Nori. 2013b. A Data Driven Approach for Algebraic Loop Invariants. In *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings.* 574–592. https://doi.org/10.1007/978-3-642-37036-6_31

Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken, and Aditya V. Nori. 2013a. Verification as Learning Geometric Concepts. In *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings.* 388–411.

Rahul Sharma, Aditya V. Nori, and Alex Aiken. 2012. Interpolants as Classifiers. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings.* 71–87. https://doi.org/10.1007/978-3-642-31424-7_11

Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. 2000. Checking Safety Properties Using Induction and a SAT-Solver. In *Formal Methods in Computer-Aided Design, Third International Conference, FMCAD 2000, Austin, Texas, USA, November 1-3, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1954)*, Warren A. Hunt Jr. and Steven D. Johnson (Eds.). Springer, 108–125. https://doi.org/10.1007/3-540-40922-X_8

Christopher Umans. 2001. The Minimum Equivalent DNF Problem and Shortest Implicants. *J. Comput. Syst. Sci.* 63, 4 (2001), 597–611. https://doi.org/10.1006/jcss.2001.1775

Caterina Urban. 2015. *Static analysis by abstract interpretation of functional temporal properties of programs.* Ph.D. Dissertation. Paris, Ecole normale supérieure.

Douglas H Wiedemann. 1987. *Hamming geometry.* Ph.D. Dissertation. University of Waterloo.