The Reachable Simulation Problem

PIERRE GANTY, IMDEA Software Insititute, Spain
NICOLAS MANINI, IMDEA Software Insititute, Spain and Universidad Politécnica de Madrid, Spain
FRANCESCO RANZATO, Università di Padova, Italy

We investigate the problem of computing the reachable blocks of the simulation equivalence and its natural counterpart for the simulation preorder, referred to as the *reachable simulation problem*. Through a theoretical investigation of this problem, we unveil a sharp contrast with the already settled case of bisimulation equivalence. Then, we design algorithms to solve the reachable simulation problem by leveraging the idea of interleaving reachability and simulation computation while possibly avoiding the computation of all the reachable states or the whole simulation preorder. Specifically, we propose algorithms achieving different guarantees on the precision of the output, and a symbolic algorithm that operates on state partitions and relations between their blocks, which is particularly well-suited for processing infinite-state systems.

CCS Concepts: • **Theory of computation → Design and analysis of algorithms**; **Concurrency**; **Program verification**; **Program analysis**.

Additional Key Words and Phrases: Simulation, Reachability, Reachable Simulation Problem, Simulation algorithm, Symbolic Data Structure

ACM Reference Format:
Pierre Ganty, Nicolas Manini, and Francesco Ranzato. 2021. The Reachable Simulation Problem. *ACM Trans. Comput. Logic* 1, 1, Article 1 (January 2021), 35 pages. https://doi.org/10.1145/3462673

1 Introduction

Given a possibly infinite labeled transition system $\mathcal{S}$, we study the problem of computing the reachable blocks of the simulation partition P_{sim} on $\mathcal{S}$, and the natural extension of this problem to the simulation preorder R_{sim} on $\mathcal{S}$. Given a state x, the upward closure of $\{x\}$ w.r.t. the relation R_{sim}, i.e. the set $R_{\text{sim}}(x)$ of states that simulate x, is called the principal of x (this term comes from the well-known notion of principal ideal [Gratzer 2011, Chapter I, Section 3.4], detailed definitions will be given in Section 3). Principals play the role of blocks when moving from partitions to preorders, and therefore the second problem we investigate in this work is that of computing the set of reachable principals of R_{sim}. By reachable, we mean the principals $R_{\text{sim}}(x)$ of the simulation preorder (resp., blocks $P_{\text{sim}}(x)$ of the simulation partition) that intersect the reachable states of $\mathcal{S}$, therefore ignoring unreachable principals and blocks, which are typically of no/negligible interest. A naïve solution to this problem—that we call the *reachable simulation problem*—would be: first, compute the simulation preorder (partition), and then, filter out the principals (blocks) containing no reachable states. However, this would require the computation of the entire simulation preorder and, possibly, of all the reachable states. Here, we present a completely different solution relying

Authors' Contact Information: Pierre Ganty, IMDEA Software Insititute, Pozuelo de Alarcón, Spain, pierre.ganty@imdea.org; Nicolas Manini, IMDEA Software Insititute, Pozuelo de Alarcón, Spain and Universidad Politécnica de Madrid, Madrid, Spain, nicolas.manini@imdea.org; Francesco Ranzato, Università di Padova, Padova, Italy, francesco.ranzato@unipd.it.

ACM 1557-945X/2021/1-ART1
https://doi.org/10.1145/3462673

on a convoluted interleaving of reachability and simulation computation, possibly avoiding the computation of all the reachable states and of the full simulation preorder relation R_{sim}.

Contributions. In Section 4 we define the reachable simulation problem and show, through an unsolvability proof (cf. Section 4.1), that there is stark contrast w.r.t. the problem of computing the reachable blocks of the bisimulation partition, settled by Lee and Yannakakis [1992] in STOC 1992. We also highlight that the reachable simulation problem is a more complex problem w.r.t. its bisimulation counterpart, even in the case of finite transition systems, where both problems can be solved. These insights allow us to show that the reachable simulation problem is a substantially harder problem w.r.t. the one tackled by Lee and Yannakakis [1992].

In Section 5 we introduce our solution to the reachable simulation problem, by putting forward an algorithm which computes the reachable part of the simulation preorder, and yields an over-approximation for the partition. Let us remark that, since we show that the reachable simulation problem is generally unsolvable, no general solution can be formulated, thus our algorithms are not guaranteed to terminate on all input systems. We prove correctness and termination on finite state systems, and extend correctness, under a simple assumption, to infinite state systems as well. Moreover, we provide examples showing termination on some infinite state systems.

In Section 6 we discuss two different kind of approximation strategies that can be adopted in solving the reachable simulation problem. We then introduce a second algorithm (cf. Section 6.3) which adopts a different approximation approach w.r.t. the previous algorithm to obtain a more precise characterization of the reachable simulation partition blocks. This algorithm, too, is proved correct and terminating on finite systems, and correctness extends to infinite state systems under an easy assumption.

Section 7 introduces the so-called 2PR triples, that is, a data structure suitable to represent relations on infinite sets. 2PR triples are used in Section 8 to design a symbolic algorithm for the reachable simulation problem. The symbolic algorithm comes with a region algebra-based representation (cf. Section 8.1) which enables an effective implementation of the algorithm. Besides inheriting the correctness guarantees of our first algorithm, we show that the 2PR-based algorithm terminates faster and more often for infinite state systems. In particular, we prove that the 2PR-based algorithm terminates on all systems having a finite bisimulation partition or when a local finiteness condition is satisfied. These termination results come with a runtime upper bound which is quadratic in the number of blocks of (a portion of) the bisimulation partition. The 2PR-based algorithm is illustrated by examples showing its behaviour.

Applications. Computing reachable simulation principals and blocks has several practical applications. A noteworthy use case of the reachable principals of the simulation preorder is given by the determinization algorithms $\textsc{Subset}(f)$ and $\textsc{Transset}(f)$ for nondeterministic finite automata designed by van Glabbeek and Ploeger [2008]. Using the simulation preorder computationally enhances these procedures (f is picked to account for the simulation preorder). In particular, only the reachable principals are used, since the automaton determinization proceeds forward starting from the initial states. It turns out that these simulation-based algorithms compute smaller deterministic automata compared to their plain versions [van Glabbeek and Ploeger 2008]. In a different context, the reachable blocks of the simulation partition define the states of the reduced quotiented transition system. The question of computing the transitions between the blocks of the reduced system has been investigated in depth by Bustan and Grumberg [2003], who explore the difference and trade-off of the $\exists\exists$ (i.e., $B \to^{\exists\exists} B'$ iff $\exists s \in B.\, \exists s' \in B'.\, s \to s'$) and $\forall\exists$ definitions (i.e., $B \to^{\forall\exists} B'$ iff $\forall s \in B.\, \exists s' \in B'.\, s \to s'$).

Furthermore, solutions to the reachable simulation problem have potential applications in program and hybrid systems verification, as past research [Gulavani et al. 2006; Majumdar et al.

2020; Pasareanu et al. 2005; Yannakakis and Lee 1997] has leveraged solutions to the reachable bisimulation problem.

Related Work. The closest work to ours is that by Lee and Yannakakis [1992], who first designed a complex interleaving of reachability and bisimulation computation, here referred to as the LY algorithm. Even though their work is highly cited, applied, and has been revisited several times (e.g.,[Alur and Henzinger 1999; Fisler and Vardi 2002]), it remains an elaborate algorithm with hidden subtleties. Moreover, for some results claimed in the original paper [Lee and Yannakakis 1992], we could not rediscover their proof arguments (either searching online, contacting authors, or trying by ourselves). To put the LY algorithm in its historical context, it was one of several algorithms to compute the reachable part of the bisimulation-based quotiented transition system [Bouajjani et al. 1991, 1992; Lee and Yannakakis 1992]. These algorithms share the interleaving of the bisimulation computation—which is a partition refinement algorithm—with the computation determining block reachability. Interleaving reachability and bisimulation computation is remarkably interesting because the resulting algorithms terminate at least as often (and possibly more often) than the naïve procedure consisting in first computing the bisimulation and next determining its reachable blocks. Later on, Alur and Henzinger [1999] revisited these algorithms for reachable bisimulation in their unpublished book on computer-aided verification [Alur and Henzinger 1999, Chapter 4], as well as the theoretical and experimental comparison made by Fisler and Vardi [2002]. We also mention that algorithms combining reachability and bisimulation computation inspired by the LY algorithm have been used in several different contexts ranging from program analysis [Gulavani et al. 2006; Pasareanu et al. 2005] to hybrid systems verification, where Majumdar et al. [2020] employ a LY-like approach for language preserving minimization for controller design.

We focus on simulation since it provides a better state space reduction than bisimulation, while retaining enough precision for checking all linear temporal logic formulas or branching temporal logic formulas without quantifier switches [Bensalem et al. 1992; Bustan and Grumberg 2003; Clarke et al. 2018; Grumberg and Long 1991, 1994; Loiseaux et al. 1995]. Moreover, infinite state systems like 2D rectangular automata may have infinite bisimilarity quotients, yet they always have finite similarity quotients (see [Henzinger et al. 1995; Henzinger and Kopke 1995]). There is a large body of work [Bloom and Paige 1995; Cécé 2017; Crafa et al. 2011; Gentilini et al. 2003; Glabbeek and Ploeger 2008; Henzinger et al. 1995; Ranzato 2013, 2014; Ranzato and Tapparo 2007, 2010; Tan and Cleaveland 2001] on efficiently computing the simulation preorder, through both explicit or symbolic algorithms. Kucera and Mayr [2002a,b] compared simulation and bisimulation equivalence using their computational complexity, and justified the claim that similarity is computationally harder than bisimilarity.

To the best of our knowledge, no previous work considered the problem of computing the reachable principals of the simulation preorder or the reachable blocks of the simulation partition. This is an extended and revised version of the conference paper [Ganty et al. 2024].

2 Motivating Example

In this section, we provide an example introducing the challenges of the reachable simulation problem. This example will then be used to give an intuitive explanation of our solution, and, later on, to guide the reader through an execution of our algorithm.

Consider the family of *infinite-state* transition systems parameterized over an integer value $k \geq 0$, as depicted in Figure 1. For every value k, the set of states is $\mathbb{Z} \cup \{0', \ldots, k'\}$, and the transition relation between states is given by the arrows as shown in the diagram of Figure 1. The initial state is $0'$, denoted by the incoming blue arrow. Consider the initial partition of states given by the three

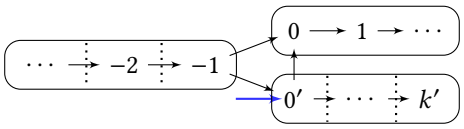


Fig. 1. A parameterized infinite-state transition system.

blocks $\{n \in \mathbb{Z} \mid n < 0\}$, $\{n \in \mathbb{Z} \mid n \geq 0\}$ and $\{n' \mid 0 \leq n \leq k\}$, depicted as solid boxes surrounding the states.

The goal is that of computing the part of the simulation relation (with respect to the initial partition) which involves reachable states in the system. More specifically, we want to compute only the portion (i.e., pairs of states) of the simulation preorder which involves reachable states, therefore possibly avoiding the computation of the whole relation.

Simulation Preorder. To illustrate the advantage of avoiding computations over unreachable states, let us first analyze the simulation preorder R_{sim} over the whole state space, as induced by the initial partition. It turns out that each state $n \geq 0$ can be simulated by any other state identified by a non-negative number, that is, each pair (n, m) such that $n, m \geq 0$ belongs to the simulation preorder. Moreover, each state n' can be simulated by all the states m' such that $0 \leq m \leq n$ holds. That is, each pair (n', m') where $0 \leq m \leq n \leq k$ holds, belongs to the simulation preorder. Additionally, each state $n < -1$ can be simulated by any other state $m \leq n$, that is, each pair (n, m) such that $m \leq n < -1$ holds, belongs to the simulation preorder. Finally, the state -1 can be simulated by itself only, thus $(-1, -1)$ is in the simulation preorder.

It is easily seen that computing the exact simulation preorder R_{sim} through the usual iterative relation refinement approaches—namely, starting from the initial partition and removing pairs not contained in the simulation preorder via an unstability check, as will be illustrated by Algorithm 5 in Section 9—requires the removal of an infinite amount of pairs from the initial partition. In particular, it would require updating the simulation information for an infinite amount of states, meaning that any relation refinement procedure following this approach would not terminate. In fact, each of the states $n < -1$ needs to have its simulation information updated, since initially all the negative states are candidate to simulate n (i.e., every pair (n, m), such that $n, m \leq -1$ belongs in the initial partition), but only those smaller than n can actually simulate it (e.g., every pair (n, m) such that $-1 \geq n < m$ is not in the simulation preorder and therefore needs to be removed from the relation through a refinement).

Our Approach. Let us remark three observations about each transition system in this family and their simulation equivalence classes, where the simulation equivalence class of any state x is the set of states which can be simulated by exactly the same states as x, that is, the classes of the equivalence induced by the simulation preorder. (1) there are infinitely many reachable states, namely, the states in $\{n \in \mathbb{Z} \mid n \geq 0\} \cup \{n' \mid n \in \mathbb{Z},\ 0 \leq n \leq k\}$ are all reachable from the initial state $0'$; (2) there are infinitely many simulation equivalence classes, depicted by dotted lines splitting the boxes of the initial partition; and (3) yet, there are finitely many simulation equivalence classes that are reachable, i.e., finitely many dotted blocks in the diagram include reachable states.

In this work, we tackle the challenge of effectively computing information such as in (3), that is, simulation equivalence blocks, and/or simulation information, for reachable states. Because of points (1) and (2) we must rule out naïve solutions that would include a computation of all the reachable states—a simple reachability computation will not terminate—or refining the initial

partition up to the simulation preorder/partition—a simulation algorithm would not terminate because the simulation partition is infinite.

We observe that, for each system in our family, each state $n \geq 0$ can be simulated by exactly the states which are in its initial partition's block, thus meaning that its simulation information does not need to be refined. Moreover, let us notice that only a *finite* set of states, i.e. $\{n' \mid n \in \mathbb{Z},\ 0 \leq n < k\}$, needs to have its simulation information updated w.r.t. the initial partition.

Intuitively, our reachable simulation algorithm can terminate on such inputs by interleaving a partial exploration of the reachable state space and using that information to guide the refinement process so as to execute finitely many relation refinements. More precisely, our algorithm will also avoid the computation of the whole infinite set of reachable states by only exploring new states which belong to not-yet-explored blocks of simulation equivalent states.

Following this intuition, we define algorithms alternating bounded state space exploration and relation refinement, which can indeed effectively compute information such as (3), while avoiding the pitfalls of computing all the reachable states or computing the full simulation partition.

Section 5.3 will show in detail a full run of our algorithm on this example.

3 Background

Orders and Partitions. Given a (possibly infinite) set Σ, we denote with $\wp(\Sigma)$ the powerset of Σ, and with $\mathrm{Rel}(\Sigma) \triangleq \wp(\Sigma \times \Sigma)$ the set of relations over Σ. If $R \in \mathrm{Rel}(\Sigma)$ then: for $S \subseteq \Sigma$, $R(S) \triangleq \{s' \in \Sigma \mid \exists s \in S.\, (s, s') \in R\}$; for $s \in \Sigma$, the set $R(s) \triangleq R(\{s\})$ is the *principal* of s; $R^{-1} \triangleq \{(y, x) \in \Sigma \times \Sigma \mid (x, y) \in R\}$ is the converse relation of R. Moreover, for a given set $S \subseteq \Sigma$, we denote by $R^S \triangleq \{R(x) \in \wp(\Sigma) \mid x \in \Sigma, R(x) \cap S \neq \emptyset\}$ the set of principals of R that intersect S. A relation $R \in \mathrm{Rel}(\Sigma)$ is a preorder if it is reflexive and transitive, and $\mathrm{PreO}(\Sigma) \triangleq \{R \in \mathrm{Rel}(\Sigma) \mid R \text{ is a preorder}\}$ denotes the set of preorders on Σ. Moreover, $R \in \mathrm{Rel}(\Sigma)$ is an equivalence on Σ if it is a symmetric preorder. A partition of Σ consists of pairwise disjoint nonempty subsets of Σ, called blocks, whose union is Σ, and $\mathrm{Part}(\Sigma)$ denotes the set of partitions of Σ. We consider finite partitions (i.e., consisting of finitely many blocks), unless otherwise specified. It is well known that a partition defines an equivalence relation, and vice versa, where blocks of the partition and equivalence classes coincide. Hence, given a partition $P \in \mathrm{Part}(\Sigma)$, $P(s)$, $P(S)$ and P^S (for $S \subseteq \Sigma, s \in \Sigma$) are well-defined thanks to the equivalence defined by P. In particular, $P(s)$ is the block including s, $P(S) = \cup\{P(s) \in P \mid s \in S\}$, and $P^S = \{P(s) \in P \mid s \in S\} \in \mathrm{Part}(P(S))$. Given two partitions $P, Q \in \mathrm{Part}(\Sigma)$, P is coarser than Q, denoted by $Q \preceq P$, if the equivalence relation underlying Q is a subset of the underlying equivalence of P.

Induced Partitions. More generally, any relation $R \in \mathrm{Rel}(\Sigma)$ (not necessarily an equivalence or a preorder) induces a partition of Σ defined as $\{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma}$. Two elements belong to the same block of the induced partition if their image by R coincide. This general definition of partition induced by a relation has the following desirable properties. When R is an equivalence relation then the blocks of its induced partition coincide with the equivalence classes of R. Moreover, if R is a preorder then the blocks of its induced partition coincide with the equivalence classes of the equivalence relation $R \cap R^{-1}$. These induced partitions will be a key ingredient of our approach and will be used throughout the paper.

Simulation and Bisimulation. Let $G = (\Sigma, I, L, \rightarrow)$ be a (labeled) transition system (TS), where Σ is a (possibly infinite yet countable) set of states, $I \subseteq \Sigma$ are the initial states, L is a finite set of action labels, and $\rightarrow \subseteq \Sigma \times L \times \Sigma$ is the labeled transition relation, where we denote $(x, a, y) \in \rightarrow$ as $x \xrightarrow{a} y$. When L is a singleton set or when the label is unimportant we simply write $x \rightarrow y$. This comes handy in our examples where we assume L is a singleton. Given $a \in L$, $\mathrm{post}_a \colon \wp(\Sigma) \rightarrow \wp(\Sigma)$ denotes the

usual successor transformer $\mathrm{post}_a(X) \triangleq \{y \in \Sigma \mid \exists x \in X.\, x \xrightarrow{a} y\}$, and, dually, $\mathrm{pre}_a \colon \wp(\Sigma) \to \wp(\Sigma)$ is the predecessor $\mathrm{pre}_a(Y) \triangleq \{x \in \Sigma \mid \exists y \in Y.\, x \xrightarrow{a} y\}$. Moreover, we define $\mathrm{post} \colon \wp(\Sigma) \to \wp(\Sigma)$ as $\mathrm{post}(X) \triangleq \cup_{a \in L} \mathrm{post}_a(X)$ and, symmetrically, $\mathrm{pre} \colon \wp(\Sigma) \to \wp(\Sigma)$ as $\mathrm{pre}(X) \triangleq \cup_{a \in L} \mathrm{pre}_a(X)$. Thus, $\mathrm{post}^*(I) \triangleq \cup_{n \in \mathbb{N}} \mathrm{post}^n(I)$ is the set of *reachable states*.

Given an (initial) preorder $R_i \in \mathrm{PreO}(\Sigma)$, a relation $R \in \mathrm{Rel}(\Sigma)$ is a *simulation* on G w.r.t. R_i if: (1) $R \subseteq R_i$; (2) $(s,t) \in R$ and $s \xrightarrow{a} s'$ imply $\exists t'.\ t \xrightarrow{a} t'$ and $(s',t') \in R$. Given two principals $R(s)$, $R(s')$ such that $s \xrightarrow{a} s'$, we define that $R(s)$ is *a-stable* (or simply *stable*) w.r.t. $R(s')$ when $R(s) \subseteq \mathrm{pre}_a(R(s'))$ holds, otherwise $R(s)$ is called *a-unstable* (or simply *unstable*) w.r.t. $R(s')$, and, in this case, $R(s')$ can *refine* $R(s)$ to $R(s) \cap \mathrm{pre}_a(R(s'))$. As a consequence, point (2) in the above simulation definition is equivalent to: (2′) for every transition $s \xrightarrow{a} s'$ in G, $R(s)$ is a-stable w.r.t. $R(s')$. The greatest (w.r.t. $\subseteq$) simulation relation on G exists and turns out to be a preorder called the *simulation preorder* of G w.r.t. R_i, denoted by $R_{\mathrm{sim}} \in \mathrm{PreO}(\Sigma)$. We denote by $P_{\mathrm{sim}} \in \mathrm{Part}(\Sigma)$ the partition induced by R_{sim} and call it the *simulation partition* (or similarity). Observe that since R_{sim} is a preorder, we have that $P_{\mathrm{sim}} \in \mathrm{Part}(\Sigma)$ coincides with the equivalence classes of the similarity equivalence $R_{\mathrm{sim}} \cap (R_{\mathrm{sim}})^{-1}$. A relation $R \in \mathrm{Rel}(\Sigma)$ is a *bisimulation* on G w.r.t. an (initial) partition $P_i \in \mathrm{Part}(\Sigma)$ if both R and R^{-1} are simulations on G w.r.t. P_i. The greatest (w.r.t. $\subseteq$) bisimulation relation on G w.r.t. P_i exists, and turns out to be an equivalence called *bisimulation equivalence* (or bisimilarity), denoted by R_{bis}. The partition $P_{\mathrm{bis}} \in \mathrm{Part}(\Sigma)$ induced by R_{bis} is called the *bisimulation partition*.

On the Initial Preorder. We point out that the role of an initial preorder R_i is that of having some a priori simulation information, e.g., accepting states in a finite state automaton simulate non-accepting ones but not the other way around [van Glabbeek and Ploeger 2008]. For the bisimulation case, such a priori information is conveyed by an equivalence—e.g. specified via a labelling over the state space like in Kripke structures—and the natural generalization of the initial equivalence to the simulation case is an initial preorder. Nevertheless, as mentioned above, an equivalence relation can be used as R_i too, since it is a particular case of a preorder relation.

4 The Problem of Computing Reachable Simulations

We define the problem investigated in this work, which extends in a natural way the reachable bisimulation problem tackled by Lee and Yannakakis [1992].

*Problem 4.1 (**The Reachable Simulation Problem**).*
Given: A labeled transition system $G = (\Sigma, I, L, \to)$ and an initial preorder $R_i \in \mathrm{PreO}(\Sigma)$.
Compute: The reachable principals of the simulation preorder R_{sim}, w.r.t. R_i, and the reachable blocks of the simulation partition P_{sim}.

A first matter we face is related to the notion of reachability. The notion of reachable blocks (rb) for any partition $P \in \mathrm{Part}(\Sigma)$, such as P_{sim}, is (trivially) defined as $P^{\mathrm{post}^*(I)}$, and the following equality follows from the definition:

$$P^{\mathrm{post}^*(I)} = \{P(s) \in P \mid s \in \Sigma,\ P(s) \cap \mathrm{post}^*(I) \neq \varnothing\} = \{P(s) \in P \mid s \in \mathrm{post}^*(I)\}\ . \tag{rb}$$

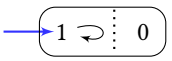
Moving from bisimulation to simulation, (rb) can be generalized to any of the two following definitions of *reachable principals* (rp) of a reflexive relation $R \in \mathrm{Rel}(\Sigma)$, which can both be deemed adequate:

$$R^{\mathrm{post}^*(I)} = \{R(s) \in \wp(\Sigma) \mid s \in \Sigma,\ R(s) \cap \mathrm{post}^*(I) \neq \varnothing\}\ , \tag{rp$_1$}$$

$$R^{\mathrm{post}^*(I)}_{\mathrm{alt}} \triangleq \{R(s) \in \wp(\Sigma) \mid s \in \mathrm{post}^*(I)\}\ . \tag{rp$_2$}$$

Clearly, when R is an equivalence relation, (rp_1) and (rp_2) coincide and boil down to (rb). However, in general, only $R_{\mathrm{alt}}^{\mathrm{post}^*(I)} \subseteq R^{\mathrm{post}^*(I)}$ holds and, moreover, the inclusion $(R_{\mathrm{sim}})_{\mathrm{alt}}^{\mathrm{post}^*(I)} \subsetneq (R_{\mathrm{sim}})^{\mathrm{post}^*(I)}$ can hold strictly, and, for some systems, $R^{\mathrm{post}^*(I)}$ could even be infinite whilst $R_{\mathrm{alt}}^{\mathrm{post}^*(I)}$ be a finite set. Let us show a very simple example where this inclusion holds strictly.

Example 4.2. Consider the transition system depicted below, the initial preorder $R_i = \{0,1\}\times\{0,1\}$, and the block induced by R_i, represented as a box in the diagram. The simulation preorder w.r.t. R_i is given by $R_{\mathrm{sim}}(0) = \{0,1\}$, $R_{\mathrm{sim}}(1) = \{1\}$, and dotted lines in the diagram delimit the blocks of the simulation partition P_{sim}.



The set of reachable states is $\mathrm{post}^*(I) = \{1\}$, and thus we get that the set of reachable principals as per (rp_2) is the singleton $\{R_{\mathrm{sim}}(1)\}$, while the set of reachable principals as per (rp_1) is $\{R_{\mathrm{sim}}(0), R_{\mathrm{sim}}(1)\}$, which is therefore a strict superset. ◇

4.1 Unsolvability of the Reachable Simulation Problem

Problem 4.1 is, in general, unsolvable—i.e., no algorithm exists for computing the reachable principals and blocks of, resp., R_{sim} and P_{sim}—even under the assumption that P_{sim} is a finite partition. In particular, we observe that the subtask of Problem 4.1 involving only the reachable blocks of P_{sim} is unsolvable as well. This result shows a difference with the problem of computing reachable blocks of P_{bis}, which [Lee and Yannakakis 1992, Theorem 3.1 and the following paragraph therein] proved solvable for finite bisimulations, i.e., when P_{bis} is a finite partition.

Theorem 4.3 (**Unsolvability of the Reachable Simulation Problem**). *Problem 4.1 is unsolvable, even under the assumption that P_{sim} is a finite partition.*

Theorem 4.3 can be proved by showing that solvability of Problem 4.1 would imply the decidability of the well-known undecidable halting problem for 2-counter machines. In fact, with a few termination-preserving transforms, we can characterize the halting states for a 2-CM as a block of P_{sim}. It is worth noting that the halting problem for 2-counter machines is commonly used for proving related, yet different, undecidability results about simulation [Kučera and Jančar 2006].

The reader can find the full proof, along with the necessary definitions related to 2-counter machines, in the following Section 4.2.

4.2 Proof of Theorem 4.3

We first observe that Problem 4.1 can be studied in terms of its (possible) subtasks:

(P1-b) Compute: The reachable blocks $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$.

(P1-s_1) Compute: The reachable principals $R_{\mathrm{sim}}^{\mathrm{post}^*(I)}$, according to ($\mathrm{rp}_1$).

(P1-s_2) Compute: The reachable principals $(R_{\mathrm{sim}})_{\mathrm{alt}}^{\mathrm{post}^*(I)}$, according to ($\mathrm{rp}_2$).

In particular, solvability of Problem 4.1 implies solvability of task (P1-b) and (at least one among) tasks (P1-s_1) and (P1-s_2) depending on whether (rp_1) or (rp_2) is considered as reachability notion for principals. We show that task (P1-b) (namely, computing the subset of blocks $P_{\mathrm{sim}}^{\mathrm{post}^*(I)} = \{B \in P_{\mathrm{sim}} \mid B \cap \mathrm{post}^*(I) \neq \emptyset\}$), required by both formulations of Problem 4.1, over an infinite transition system is, in general, unsolvable even under the assumption that P_{sim} is a finite partition. In particular, we show how an algorithm solving (P1-b), at least on cases where P_{sim} is finite, could be used to decide the (undecidable) halting problem for 2-counter machines—actually, an algorithm solving

the simpler task of computing just the number of reachable blocks $|P_{\text{sim}}^{\text{post}^*(I)}|$ would suffice in our proof, as we explain below.

Let us recall the necessary definitions. A 2-counter machine (2-CM) is a tuple $M = (Q, \Delta, q_0, H)$, where: Q is a finite set of states including an initial state q_0, $H \subseteq Q$ is a set of halting states with $q_0 \notin H$, and Δ is a set of transitions. The two counters are c_1 and c_2, and store non-negative integer values (and nothing else). The set $\Delta \subseteq Q \times (\{1,2\} \times \{+,-,0\}) \times Q$ comprises three types of transitions:

(1) $q \xrightarrow{c_i := c_i + 1} q'$, which increases counter c_i where $i \in \{1,2\}$;
(2) $q \xrightarrow{c_i := c_i - 1} q'$, which decreases counter c_i where $i \in \{1,2\}$; if the value of counter c_i is zero then the transition cannot be fired;
(3) $q \xrightarrow{c_i = 0} q'$, which performs a zero-test on counter c_i where $i \in \{1,2\}$; if the value of counter c_i is non-zero then the transition cannot be fired.

A configuration of M is a triple $(q, j_1, j_2) \in Q \times \mathbb{N}^2$, where $q \in Q$ is a state and $j_1, j_2 \in \mathbb{N}$ are the values stored by the counters c_1 and c_2 of M. A configuration (q, j_1, j_2) is halting when $q \in H$. The operational semantics of a 2-CM is determined by a transition relation $\rightarrow$ between configurations and is defined as expected. The 2-CM M halts on input (j_1, j_2) if there exist configurations $conf_1, \ldots, conf_k$ such that: $conf_1 \rightarrow \cdots \rightarrow conf_k$, where $conf_1 = (q_0, j_1, j_2)$ and $conf_k$ is halting. The halting problem for 2-CMs is well-known to be undecidable [Minsky 1967, Chapter 14].

*Problem 4.4 (**Halting Problem of 2-CMs**).*
Given: A 2-CM M.
Decide: Can M halt on input $(0,0)$?

Consider an instance M of the halting problem for 2-CMs. We first define a counter machine M_1 that is obtained by adding to M a new non-halting state q_1 for each non-halting state q of M. Besides the transitions of M, M_1 has three additional transitions for each new state q_1 added at the previous step: $q \xrightarrow{c_1 := c_1 + 1} q_1$, $q_1 \xrightarrow{c_1 := c_1 - 1} q$, and $q_1 \xrightarrow{c_1 = 0} q$. This definition of M_1 ensures that every non-halting configuration in $(Q \setminus H) \times \mathbb{N}^2$ can always progress to a non-halting successor configuration in $(Q \setminus H) \times \mathbb{N}^2$. Observe that adding such states and transitions does not modify whether M halts: in fact, M halts iff M_1 halts. Furthermore, we assume, without loss of generality, that halting states have no outgoing transitions. This assumption can be enforced (if needed) because if a halting state $h \in H$ has outgoing transitions then we define M_2 by duplicating in M_1 the state h and all its ingoing and outgoing transitions into a new non-halting state q_h, and, then, we remove all the outgoing transitions out of h. Again, this transformation does not modify whether M halts: M halts iff M_2 halts. We thus consider the transition system induced by M_2 with configurations $Q \times \mathbb{N}^2$ and with a singleton set of initial states $I \triangleq \{(q_0, 0, 0)\}$. Next, let us define the preorder r_i to be $(H \times Q) \cup \big((Q \setminus H) \times (Q \setminus H)\big) \in \text{PreO}(Q)$. Hence, define $R_i \in \text{PreO}(Q \times \mathbb{N}^2)$ as follows:

$$R_i \triangleq \big((H \times \mathbb{N}^2) \times (Q \times \mathbb{N}^2)\big) \cup \big(((Q \setminus H) \times \mathbb{N}^2) \times ((Q \setminus H) \times \mathbb{N}^2)\big) \ . \tag{1}$$

We show that R_i is the simulation preorder, i.e. $R_i = R_{\text{sim}}$, because we can prove that for every transition $conf_x \rightarrow conf_y$ the inclusion $R_i(conf_x) \subseteq \text{pre}(R_i(conf_y))$ holds. Firstly, note that if $conf_x$ is halting then $conf_x \rightarrow conf_y$ for no configuration $conf_y$ (the definition of M_2 guarantees this property). If $conf_x$ is non-halting then we have that $R_i(conf_x) = (Q \setminus H) \times \mathbb{N}^2$. If $conf_y$ is halting then we have $R_i(conf_y) = Q \times \mathbb{N}^2$, which, together with the fact that every configuration of $(Q \setminus H) \times \mathbb{N}^2$ has an outgoing transition (this is ensured by M_1), shows that the inclusion $R_i(conf_x) \subseteq \text{pre}(R_i(conf_y))$ holds. The last case to consider is when both $conf_x$ and $conf_y$ are

non-halting, which implies that $R_i(conf_x) = R_i(conf_y) = (Q \setminus H) \times \mathbb{N}^2$. Here, we find that the inclusion $R_i(conf_x) \subseteq \mathrm{pre}(R_i(conf_y))$ holds since every element of $(Q \setminus H) \times \mathbb{N}^2$ has an outgoing transition into some non-halting configuration.

Therefore, $R_i = R_{\mathrm{sim}}$ holds, and the simulation partition P_{sim} induced by R_{sim} is:

$$P_{\mathrm{sim}} = \{H \times \mathbb{N}^2, (Q \setminus H) \times \mathbb{N}^2\} \ . \tag{2}$$

Now, suppose that an algorithm $\mathcal{A}$ exists, which, given a TS (e.g., the system underlying a 2-CM, represented through the 2-CM) and an initial preorder R_i (e.g., encoded through r_i above), outputs, when P_{sim} is finite, the set of reachable P_{sim} blocks, that is: $\mathcal{A}(\langle \Sigma, I, L, \rightarrow \rangle, R_i) = P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$. Equivalently, one could assume, instead, that $\mathcal{A}$ outputs the number of reachable blocks, as we anticipated above. Then, we could run the algorithm $\mathcal{A}$ on the TS underlying M_2 above and R_i defined in (1), and obtain $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$, where P_{sim} is the partition defined in (2), that is,

$$\mathcal{A}(M_2, R_i) = \left\{B \in \{H \times \mathbb{N}^2, (Q \setminus H) \times \mathbb{N}^2\} \mid B \cap \mathrm{post}^*(I) \neq \emptyset\right\} \ .$$

We now observe that $\mathcal{A}(M_2, R_i)$ always contains the block $(Q \setminus H) \times \mathbb{N}^2$, since the initial state $(q_0, 0, 0)$ belongs to it. Moreover, it holds that $\mathcal{A}(M_2, R_i)$ will contain the block $H \times \mathbb{N}^2$ if and only if $\mathrm{post}^*(\{(q_0, 0, 0)\}) \cap (H \times \mathbb{N}^2) \neq \emptyset$, that is, if and only if M_2 can halt on input $(0, 0)$. This allows us to conclude that

$$|\mathcal{A}(M_2, R_i)| = 2 \Leftrightarrow M \textit{ can halt on input } (0, 0) \ .$$

Note that we are only interested in the cardinality $|\mathcal{A}(M_2, R_i)|$ of the output, and not on the actual blocks, proving how an algorithm $\mathcal{A}$ computing the number of reachable blocks alone would suffice to show unsolvability.

Therefore, since the halting problem for 2CMs is undecidable, we conclude that such an algorithm $\mathcal{A}$ does not exist, and, consequently, task (P1-b) is unsolvable, even under the assumption that P_{sim} is finite. Finally, since solvability of Problem 4.1 implies solvability of task (P1-b), we can conclude that no algorithm solving Problem 4.1 exists. In particular, the proposed reduction considers a case where the simulation partition (i.e., the partition (2)) is finite, meaning that even under the hypothesis that P_{sim} consists of a finite set of blocks, Problem 4.1 still remains unsolvable, thus completing the proof. □

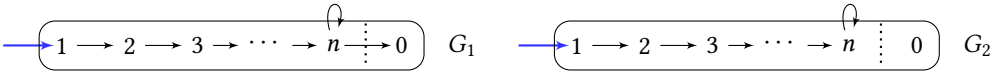
Remarks. This negative result is to be contrasted with the positive result of Lee and Yannakakis [1992] for the bisimulation partition P_{bis}, stating that the LY algorithm terminates when P_{bis} is finite [Lee and Yannakakis 1992, Theorem 3.1 and the following paragraph therein]. Intuitively, the above reduction does not work for the corresponding problem where bisimulation replaces simulation because, in general, we cannot define a finite bisimulation that splits halting and non-halting states as we did for (2). While the above proof focuses on the task (P1-b) alone, the logic behind our proof can be used as-is to show unsolvability of task (P1-s_2) too. In fact, suppose we can compute the reachable principals $R_{\mathrm{alt}}^{\mathrm{post}^*(I)}$, for the simulation relation R_{sim} defined in (1). This would lead to solve the halting Problem 4.4: the 2-CM M halts iff the principal $(Q \times \mathbb{N}^2)$ is reachable. In fact, following definition (rp_2), reachability of the principal $(Q \times \mathbb{N}^2)$ holds iff some state in $(H \times \mathbb{N}^2)$ is reachable.

4.3 Remarks for Finite State Systems

We observe a further difference between Problem 4.1 and the corresponding problem for the bisimulation case studied by Lee and Yannakakis [1992], over *finite* transition systems. The reachability problem for blocks of both P_{bis} and P_{sim} is trivially decidable for finite systems, since we can simply compute independently $\mathrm{post}^*(I)$ and P_{bis} or P_{sim}, and, check whether $\mathrm{post}^*(I) \cap B = \emptyset$ holds for every block B in P_{bis} or P_{sim}. However, for the case of bisimulation, deciding reachability of a block

$B \in P_{\text{bis}}$ can be done in $O(|P_{\text{bis}}^{\text{post}^*(I)}|)$ time by exploiting the definition of bisimulation: if x and y are bisimilar states, then x can reach (some state in) B in one transition iff y can, hence picking any single state per block of P_{bis} suffices to infer the reachable blocks from all the other states in the block. For the simulation case, deciding the reachability of $B \in P_{\text{sim}}$ is more involved: as the following example hints, to decide whether $B \in P_{\text{sim}}$ is reachable we have to check whether there is a path of arbitrary length in the transition system reaching B, and thus picking any single state per block is not enough.

Example 4.5. Let G_1, G_2 be the two transition systems depicted below (resp., left and right). Blocks of states sharing the same principal in $R_i = \mathbb{N} \times \mathbb{N}$ are represented as boxes, that is, the two boxes define the blocks of the partition induced by R_i. Moreover, dotted lines are used to delimit the blocks of the simulation partition P_{sim} w.r.t. R_i. In fact, for both systems, $R_{\text{sim}}(0) = [0, n]$, and, for all $k \in [1, n]$, $R_{\text{sim}}(k) = [1, n]$, so we get that $P_{\text{sim}} = \{[0, 0], [1, n]\}$ for G_1 and G_2.



Observe that in G_1 the block $[0, 0] \in P_{\text{sim}}$ is reachable while in G_2 it is unreachable. Hence, to decide whether $[0, 0]$ is reachable or not in these two systems, we have to detect that the state 0 is reachable in G_1 and not in G_2, and this cannot be inferred by randomly picking one state in $[1, n]$ and observing its outgoing transitions only. However, this is not the case for bisimulation, since we have that, for the system above, $P_{\text{bis}} = \{[k, k]\}_{k=0,\ldots,n}$ for G_1, while $P_{\text{bis}} = P_{\text{sim}}$ for G_2, so that the same argument cannot be applied. ◇

5 An Algorithm for Reachable Simulations

In this section we introduce our approach to the reachable simulation problem. We define Algorithm 1 which, given a system G, a preorder R_i, and an initial (possibly empty) set of reachable states σ_i, computes the reachable principals of R_{sim} according to (rp_1), and over-approximates the set of reachable blocks of P_{sim}.

This algorithm maintains a relation $R \in \text{Rel}(\Sigma)$ specified through its principals $R(x) \in \wp(\Sigma)$, and a set $\sigma \subseteq \Sigma$ of reachable states, which we call *provably reachable states*, so that $R^\sigma = \{R(x) \mid R(x) \cap \sigma \neq \emptyset\}$ are the *provably reachable principals* of R. The algorithm computes a set U of principals which can be added to R^σ by expanding σ, and the set V of unstable principal pairs, with the first element having a provably reachable principal. A principal $R(x)$ is in U if it contains an initial state or a successor of a provably reachable state. A triple $\langle a, x, x' \rangle$ is in V if the principal $R(x)$ is provably reachable and it can be refined by a principal $R(x')$, i.e., $x \xrightarrow{a} x'$ and $R(x) \not\subseteq \text{pre}_a(R(x'))$. Algorithm 1 is presented in *logical form*, meaning that in this pseudocode we do not require or provide a specific representation for the transition system G or for the sets maintained by the algorithm, namely the relation R, the provably reachable states of σ, and the sets U and V. A *symbolic* version of Algorithm 1, along with the details on the specific representations needed to implement the algorithm will be given in Section 8.

At each iteration of the while loop, Algorithm 1 either updates, in the *Search* block, the reachability information by expanding the set σ of provably reachable states, or stabilizes, in the *Refine* block, a pair of principals from V. The pseudocode of Algorithm 1 uses a nondeterministic choice between guarded commands (**nif**). We have three guarded commands: either the *Search* (lines 7–9) or the *Refine* blocks (lines 10–12) are executed, or, at line 13, when the other guards evaluate to false, the return statement is taken. Thus, every execution consists of an interleaving of *Search* and *Refine*, possibly followed by a return. Observe that the guards are such that when the algorithm terminates neither *Search* nor *Refine* are enabled.

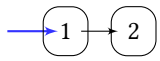
Algorithm 1: Relation-based Algorithm

Input: A system $G = (\Sigma, I, L, \to)$, an initial $R_i \in \mathrm{PreO}(\Sigma)$, an initial finite set $\sigma_i \subseteq \mathrm{post}^*(I)$.

1 $\mathrm{Rel}(\Sigma) \ni R \coloneqq R_i$;
2 $\wp(\Sigma) \ni \sigma \coloneqq \sigma_i$;
3 **while true do**
  // $\textsc{Inv}_1$: $\forall x \in \Sigma.\, R_{\mathrm{sim}}(x) \subseteq R(x) \subseteq R_i(x)$
  // $\textsc{Inv}_2$: $\sigma_i \subseteq \sigma \subseteq \mathrm{post}^*(I)$
  // $\textsc{Inv}_3$: $\forall x \in \Sigma.\, x \in R(x)$
4 $U \coloneqq \{R(x) \mid R(x) \cap \sigma = \emptyset,\ R(x) \cap (I \cup \mathrm{post}(\sigma)) \neq \emptyset\}$;
5 $V \coloneqq \{\langle a, x, x' \rangle \in L \times \Sigma^2 \mid R(x) \cap \sigma \neq \emptyset, x \xrightarrow{a} x',\ R(x) \not\subseteq \mathrm{pre}_a(R(x'))\}$;
6 **nif**
7 $(U \neq \emptyset) \longrightarrow$ *Search* :
8 **choose** $R(x) \in U,\ s \in R(x) \cap (I \cup \mathrm{post}(\sigma))$;
9 $\sigma \coloneqq \sigma \cup \{s\}$;
10 $(V \neq \emptyset) \longrightarrow$ *Refine* :
11 **choose** $\langle a, x, x' \rangle \in V$;
12 $R(x) \coloneqq R(x) \cap \mathrm{pre}_a(R(x'))$;
13 $(U = \emptyset \wedge V = \emptyset) \longrightarrow$ **return** $\langle R, \sigma \rangle$;

A principal $R(x)$ is refined at line 12 provided it is provably reachable. Upon termination, the principals in R^σ and the principals in $R_{\mathrm{sim}}^{\mathrm{post}^*(I)}$ coincide (cf. (1.a) of Theorem 5.3 below). However, R may well contain unstable principals, so that, in general, R and R_{sim} do not coincide. Moreover, line 12 might break transitivity of R, as R is not guaranteed to be a preorder during execution, and not even at termination. Turning to the simulation partition, we face a more complex situation. To start with, we provide an example showing that the partition P induced by R is such that P^σ does not coincide with $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$. In fact, P^σ might lack some of the blocks of $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$.

Example 5.1. Consider the transition system depicted below, and the initial preorder given by $R_i = \{(1,1), (2,1), (2,2)\} = R_{\mathrm{sim}}$, which coincides with the simulation preorder. The blocks induced by R_i coincide with $P_{\mathrm{sim}} = \{\{1\}, \{2\}\}$, and are represented as boxes in the diagram.

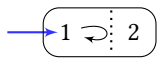


The initial state is 1, as indicated by the incoming blue arrow, and thus $\mathrm{post}^*(I) = \{1, 2\}$. Algorithm 1 on input R_i and $\sigma_i = \emptyset$ returns $R = R_{\mathrm{sim}}$ and $\sigma = \{1\}$. Therefore it follows that $P_{\mathrm{sim}}^{\mathrm{post}^*(I)} = P_{\mathrm{sim}}$, while $P^\sigma = \{\{1\}\}$, so that the inclusion $P_{\mathrm{sim}}^{\mathrm{post}^*(I)} \not\subseteq P^\sigma$ strictly holds. ◇

It turns out that Algorithm 1 aims at populating σ with just enough states to correctly characterize the reachable principals of R_{sim} (i.e., achieving equality (1.a) below), but such states are, in general, not enough to intersect all the reachable P_{sim} blocks. However, σ suffices to capture such blocks through a relaxation of the reachability notion which, in turn, induces a degree of over-approximation. This relaxed definition is given by $\{B \in P \mid R(B) \cap \sigma \neq \emptyset\}$ as defined in (1.b) below. Intuitively, according to this notion, a block is reachable if it can be simulated by any (provably) reachable state. Observe that the reachable blocks themselves are computed precisely: each block in $\{B \in P \mid R(B) \cap \sigma \neq \emptyset\}$ is a block of P_{sim}, meaning that the states contained in it are

computed in an *exact* way. In general, however, not all blocks in P belong in P_{sim}. The next example shows that the converse inclusion of (1.b) does not always hold, i.e., the containment of (1.b) may be strict.

Example 5.2. Consider the transition system below, with initial preorder $R_i = \{1,2\} \times \{1,2\}$, simulation preorder $R_{\text{sim}} = \{(1,1),(2,1),(2,2)\}$, and the induced $P_{\text{sim}} = \{\{1\},\{2\}\}$. The box depicts the single block induced by R_i and dotted lines delimit the blocks of P_{sim}.



The set of reachable states is given by $\text{post}^*(I) = \{1\}$. Algorithm 1 on input R_i and $\sigma_i = \emptyset$ outputs $R = R_{\text{sim}}$ and $\sigma = \{1\}$. Thus, the inclusion of (1.b) is strict since $\{\{1\}\} \subsetneq \{\{1\},\{2\}\}$. ◇

While we have shown that the inclusion might be strict, we point out that, however, it is not arbitrarily loose since a block $B \in P$ such that $R(B) \cap \sigma \neq \emptyset$ (cf. (1.b)) is guaranteed to be simulated by some reachable state. Therefore, R_{sim} limits the magnitude of this over-approximation.

Section 6 further investigates the nature of this approximation, and introduces Algorithm 3, which, roughly speaking, turns the inclusion of equation (1.b) into an equality with some caveats.

5.1 Correctness and Termination of Algorithm 1

We now show that our algorithm is correct (formally stating the points discussed above), and that it terminates on finite state systems.

Theorem 5.3 (**Finite Correctness of Algorithm 1**). *Let $\langle R, \sigma\rangle \in \text{Rel}(\Sigma) \times \wp(\Sigma)$ be the output of Algorithm 1 on input G with $|\Sigma| \in \mathbb{N}$, $R_i \in \text{PreO}(\Sigma)$, and $\sigma_i \subseteq \text{post}^*(I)$. Moreover, let $P \in \text{Part}(\Sigma)$ be the partition induced by R. Then:*

$$R_{\text{sim}}^{\text{post}^*(I)} = R^{\sigma} \quad , \tag{1.a}$$

$$P_{\text{sim}}^{\text{post}^*(I)} \subseteq \{B \in P \mid R(B) \cap \sigma \neq \emptyset\} \quad . \tag{1.b}$$

Proof. The following statements on the output $\langle R, \sigma\rangle$ hold:

(i) At every iteration, R is reflexive.
This holds because at the beginning R is a preorder and the update and refine block of R at lines 10–12 of Algorithm 1 preserves the reflexivity of R, in particular, the refinement statement at line 12.

(ii) At every iteration, and for all $y \in \Sigma$, $R_{\text{sim}}(y) \subseteq R(y)$.
This holds because the *Refine* block of Algorithm 1 is always correct, so that $R_{\text{sim}}(y) \subseteq R(y) \subseteq R_i(y)$ holds for every iteration of Algorithm 1.

(iii) At termination, $x \in \text{post}^*(I) \Rightarrow R(x) \cap \sigma \neq \emptyset$.
This is proven by induction on $n \in \mathbb{N}$ such that $I \to^n x$. If $n = 0$ then $x \in I$, so that, since R is reflexive following (i), $x \in R(x)$ and therefore $R(x) \cap (I \cup \text{post}(\sigma)) \neq \emptyset$. Hence, $U = \emptyset$ implies that $R(x) \cap \sigma \neq \emptyset$. If $n > 0$ then $I \to^n x' \xrightarrow{a} x$, and, by inductive hypothesis, $R(x') \cap \sigma \neq \emptyset$. Therefore, since $V = \emptyset$, $R(x') \subseteq \text{pre}_a(R(x))$ must hold. Thus, $\text{pre}_a(R(x)) \cap \sigma \neq \emptyset$, so that $R(x) \cap \text{post}_a(\sigma) \neq \emptyset$. Hence, $U = \emptyset$ implies $R(x) \cap \sigma \neq \emptyset$.

(iv) At termination, for all $x \in \Sigma$, $R(x) \cap \sigma \neq \emptyset \Rightarrow R(x) = R_{\text{sim}}(x)$.
Let Sim be the basic simulation algorithm, recalled as Algorithm 5 in Section 9, whose input is the preorder R. By (ii), $R_{\text{sim}} \subseteq R \subseteq R_i$. Thus, $\text{Sim}(R) = R_{\text{sim}}$ because $R_{\text{sim}} = \text{Sim}(R_{\text{sim}}) \subseteq \text{Sim}(R) \subseteq \text{Sim}(R_i) = R_{\text{sim}}$. Assume, by contradiction, that $\mathbb{S} \triangleq \{R(x) \in \wp(\Sigma) \mid R(x) \cap \sigma \neq \emptyset, R(x) \neq R_{\text{sim}}(x)\} \neq \emptyset$, so that each $R(x) \in \mathbb{S}$, will be refined at some iteration of $\text{Sim}(R)$.

Then, let $R(x) \in \mathbb{S}$ be the first principal in $\mathbb{S}$ such that $R(x)$ is refined by Sim at some iteration *it* whose current relation is R'. Thus, each principal $R(z) \in \mathbb{S}$ is such that $R(z) = R'(z)$ because no principal in $\mathbb{S}$ was refined by Sim before this iteration *it*. Let $R'(y) \subseteq R(y)$ be the principal of R' used by Sim in this iteration *it* to refine $R(x)$, so that $x \xrightarrow{a} y, R'(x) = R(x) \not\subseteq \mathrm{pre}_a(R'(y))$. We have that $R(x) \cap \sigma \neq \emptyset, x \xrightarrow{a} y$ and $V = \emptyset$ entail $R(x) \subseteq \mathrm{pre}_a(R(y))$. Hence, $R(x) \cap \sigma \neq \emptyset$ implies $R(y) \cap \mathrm{post}_a(\sigma) \neq \emptyset$, so that $U = \emptyset$ entails $R(y) \cap \sigma \neq \emptyset$. If $R(y) \notin \mathbb{S}$ then $R(y) \cap \sigma \neq \emptyset$ implies $R(y) = R_{\mathrm{sim}}(y) \subseteq R'(y) \subseteq R(y)$, so that $R'(y) = R(y)$ holds. If $R(y) \in \mathbb{S}$ then, since $R(x)$ is the first principal in $\mathbb{S}$ to be refined by Sim, we have that $R'(y) = R(y)$ holds. Thus, in both cases, $R'(y) = R(y)$ must hold, so that $R(x) \not\subseteq \mathrm{pre}_a(R(y))$. Moreover, $R(x) \cap \sigma \neq \emptyset$, $x \xrightarrow{a} y$ and $V = \emptyset$ imply $R(x) \subseteq \mathrm{pre}_a(R(y))$, which is therefore a contradiction. Thus, $\mathbb{S} = \emptyset$, and, in turn, $R(x) = R_{\mathrm{sim}}(x)$.

(v) At every iteration, $\sigma \subseteq \mathrm{post}^*(I)$.
This follows by $\sigma_i \subseteq \mathrm{post}^*(I)$ and because σ is updated at line 9 of Algorithm 1 by adding $s \in I \cup \mathrm{post}(\sigma)$ and $\mathrm{post}^*(I)$ is the least fixpoint of $\lambda X.I \cup \mathrm{post}(X)$.

Let us now show the equality (1.a).

($\subseteq$) Let $x \in \Sigma$ such that $R_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \emptyset$. Thus, there exists $s \in \mathrm{post}^*(I)$ such that $s \in R_{\mathrm{sim}}(x)$. By (iii), $R(s) \cap \sigma \neq \emptyset$. By (iv), $R(s) = R_{\mathrm{sim}}(s)$. Moreover, $s \in R_{\mathrm{sim}}(x)$ implies $R_{\mathrm{sim}}(s) \subseteq R_{\mathrm{sim}}(R_{\mathrm{sim}}(x)) = R_{\mathrm{sim}}(x)$, and since, by (ii), $R_{\mathrm{sim}}(x) \subseteq R(x)$, we obtain $R_{\mathrm{sim}}(s) \subseteq R(x)$. Thus, $R(s) \subseteq R(x)$ holds, so that $R(s) \cap \sigma \neq \emptyset$ entails $R(x) \cap \sigma \neq \emptyset$, and, in turn, by (iii), $R(x) = R_{\mathrm{sim}}(x)$.

($\supseteq$) Let $x \in \Sigma$ such that $R(x) \cap \sigma \neq \emptyset$. By (iv), $R(x) = R_{\mathrm{sim}}(x)$, so that $R_{\mathrm{sim}}(x) \cap \sigma \neq \emptyset$. By (v), $R_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \emptyset$.

Let us now prove (1.b). Let $x \in \Sigma$ such that $P_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \emptyset$. By definition of P, we have that for all $x \in \Sigma$, $P(x) = \{y \in \Sigma \mid R(x) = R(y)\}$. Since $P_{\mathrm{sim}}(x) \subseteq R_{\mathrm{sim}}(x)$, we have that $R_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \emptyset$. Let $y \in P_{\mathrm{sim}}(x)$. Observe that $P_{\mathrm{sim}}(x) = \{z \in \Sigma \mid R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(z)\}$. Thus, $R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(y)$, so that $R_{\mathrm{sim}}(y) \cap \mathrm{post}^*(I) \neq \emptyset$. Thus, by (1.a), $R_{\mathrm{sim}}(y) = R(y)$ and $R(y) \cap \sigma \neq \emptyset$. In particular, $R(x) = R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(y) = R(y)$. Therefore, $P_{\mathrm{sim}}(x) \subseteq \{y \in \Sigma \mid R(x) = R(y)\} = P(x)$. On the other hand, if $z \in P(x)$ then $R(z) = R(x)$, so that $R(x) \cap \sigma \neq \emptyset$ implies $R(z) \cap \sigma \neq \emptyset$. Thus, by (1.a), $R_{\mathrm{sim}}(z) = R(z)$. Hence, $R_{\mathrm{sim}}(z) = R(z) = R(x) = R_{\mathrm{sim}}(x)$, thus proving that $P(x) \subseteq P_{\mathrm{sim}}(x)$, and therefore $P(x) = P_{\mathrm{sim}}(x)$ holds. Moreover, $R(x) \cap \sigma \neq \emptyset$, thus proving (1.b). □

Theorem 5.4 (**Termination of Algorithm 1**). *Let $G = (\Sigma, I, L, \to)$ with $|\Sigma| \in \mathbb{N}$, $R_i \in \mathrm{PreO}(\Sigma)$, and $\sigma_i \subseteq \mathrm{post}^*(I)$. Then, Algorithm 1 terminates on input G, R_i, and σ_i.*

Proof. We first observe that each *Search* iteration adds some new state to σ through the update at line 9. Thus, since Σ has finitely many elements, Algorithm 1 will always execute a finite number of *Search* iterations. Similarly, executing the *Refine* block is guaranteed to remove some state from at least one principal of the current relation, and since, initially, each principal has finitely many elements, the overall number of *Refine* iterations is also finite. Therefore, since every iteration of the while-loop either executes a *Search* or a *Refine*, the algorithm terminates after a finite number of iterations. Finally, we observe that each iteration computes a finite number of operations. □

5.2 Correctness and Termination for Infinite State Systems

Theorem 5.3 is introduced on finite systems for clarity reasons when formulating the proof. Nevertheless, the proof argument of Theorem 5.3 extends to infinite state systems—i.e. the condition $|\Sigma| \in \mathbb{N}$ can be removed from the hypotheses of the theorem—when the following assumption holds:

Assumption 5.5 (ω-Convergence for Simulation Approximants). Given a transition system $G = (\Sigma, I, L, \to)$, and an initial preorder $R_i \in \mathrm{PreO}(\Sigma)$, let $\preceq_0 \triangleq R_i$ and $\preceq_n$, for $n > 0$ be the strong

simulation approximants as defined in [Hofman et al. 2016, Points (2) and (3) of Definition 30]. Then, the ω-convergence assumption holds iff $\preceq_\omega = R_{\mathrm{sim}}$, where ω is the first limit ordinal.

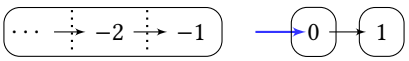
Note that Assumption 5.5 holds at least on all finitely branching systems, as stated in [Hofman et al. 2016, paragraph following Definition 30]. We can now formally state the correctness result for Algorithm 1 on possibly infinite systems as follows.

Theorem 5.6 (**Correctness of Algorithm 1**). *Let $\langle R, \sigma\rangle \in \mathrm{Rel}(\Sigma) \times \wp(\Sigma)$ be the output of Algorithm 1 on input G, $R_i \in \mathrm{PreO}(\Sigma)$ and $\sigma_i \subseteq \mathrm{post}^*(I)$. Moreover, let $P \in \mathrm{Part}(\Sigma)$ be the partition induced by R. If Assumption 5.5 holds, then conditions (1.a) and (1.b) are satisfied.*

The details of the extension of the proof to the infinite case, are presented (along with extensions for the other correctness theorems) in Section 9.

Commenting on termination, we point out that even though Theorem 5.4 guarantees termination on finite systems only, Algorithm 1 can terminate on some infinite systems too, and even when P_{sim} contains infinitely many blocks. One such case is given in the following intuitive example.

Example 5.7. Consider the depicted transition system with infinitely many states, and the initial preorder given by $R_i(0) = \{0\}$, $R_i(1) = \{0, 1\}$, and for all $n < 0$, $R_i(n) =]-\infty, -1]$. Boxes in the diagram denote the blocks of the partition induced by R_i. The simulation preorder is therefore given by: $R_{\mathrm{sim}}(0) = \{0\}$, $R_{\mathrm{sim}}(1) = \{0, 1\}$, and $R_{\mathrm{sim}}(n) =]-\infty, n]$ for each $n < 0$. Notice that R_{sim} has infinitely many principals, and the corresponding infinitely many blocks of P_{sim} are delimited by dotted lines.



After one *Search* iteration of Algorithm 1 (on input R_i, $\sigma_i = \varnothing$), we get $\sigma = \{0\}$. At this point, $V = \varnothing$ and $U = \varnothing$ holds, and the algorithm returns the correct result.
Let us consider instead the process of refining each principal $R_i(x)$ of the initial preorder such that $R_i(x) \neq R_{\mathrm{sim}}(x)$ one-by-one (this is what the basic simulation Algorithm 5 would do). This process, which converges to R_{sim}, cannot terminate after finitely many steps since infinitely many principals need to be refined. ◇

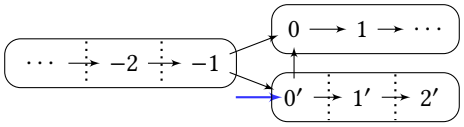
The motivating example of Section 2 is a more elaborate case of an infinite transition system with an infinite simulation partition P_{sim}, where Algorithm 1 terminates after refining some principals. Next, we provide a full execution of Algorithm 1 on the motivating example.

5.3 Execution on the Motivating Example

Let us run Algorithm 1 on the motivating example of Section 2. We fix $k = 2$ and obtain the infinite state transition system depicted below (note that a similar execution is obtained for any value k). The initial and simulation preorders are thus given by:

$$R_i(x) = \begin{cases} \mathbb{Z} \setminus \mathbb{N} & \text{if } x \in \mathbb{Z},\ x < 0 \\ \mathbb{N} & \text{if } x \in \mathbb{Z},\ x \geq 0 \\ \{0', 1', 2'\} & \text{if } x \in \{0', 1', 2'\} \end{cases} \qquad R_{\mathrm{sim}}(x) = \begin{cases} \{y \in \mathbb{Z} \mid y \leq x\} & \text{if } x \in \mathbb{Z},\ x < -1 \\ \{-1\} & \text{if } x = -1 \\ \mathbb{N} & \text{if } x \in \mathbb{Z},\ x \geq 0 \\ \{m' \mid 0 \leq m \leq n\} & \text{if } x = n',\ n \in \{0, 1, 2\} \end{cases}$$

As usual, states sharing the same principal in R_i are depicted in boxes, while dotted lines delimit the blocks of the simulation partition P_{sim}. The initial state is $0'$ and thus the set of reachable states is $\mathrm{post}^*(I) = \{0', 1', 2'\} \cup \mathbb{N}$.



We recall that all arrows in the diagram have an implicit action label (say, $a \in L = \{a\}$), and we omit them in the diagram for clarity reasons. Now we detail an execution of Algorithm 1 with input R_i and $\sigma_i = \emptyset$:

1st iteration: At the beginning, $U = \{R_i(0')\} = \{\{0', 1', 2'\}\}$, since the principals intersecting $I = \{0'\}$ are those corresponding to states $0'$, $1'$ and $2'$. Moreover, since $\sigma = \emptyset$, then $V = \emptyset$ holds. Hence, the *Search* block is executed and σ is updated: at line 9 we get $\sigma = \{0'\}$.

2nd iteration: Since $\sigma = \{0'\}$, we get $U = \{R_i(0)\} = \{\mathbb{N}\}$ because $\mathbb{N} \cap \mathrm{post}(\sigma) \neq \emptyset$. Note that $R(0')$, $R(1')$ and $R(2')$ are not in U at this point because they contain the provably reachable state $0'$. Moreover, $V = \{\langle a, 0', 1'\rangle, \langle a, 1', 2'\rangle\}$ since $\mathrm{pre}_a(R_i(1')) = \mathrm{pre}_a(R_i(2')) = \{0', 1'\}$, and $R_i(0') = R_i(1') = \{0', 1', 2'\} \not\subseteq \{0', 1'\}$. Assume that Algorithm 1 nondeterministically executes a *Search* iteration, then it will update σ by choosing the principal $R(0)$ in U: at line 9 the set of provably rechable states σ is expanded and we get $\sigma = \{0', 0\}$.

3rd iteration: We get $U = \emptyset$ since all the states in $I \cup \mathrm{post}(\sigma)$ occur in principals having a nonempty intersection with σ, therefore σ cannot be expanded so to reach a principal which is not provably reachable. Moreover, V is as in the previous iteration, that is $V = \{\langle a, 0', 1'\rangle, \langle a, 1', 2'\rangle\}$. In fact, note that $R_i(0)$, which is now provably reachable, does not induce new unstable triples in V. Assume that Algorithm 1 picks $\langle a, 1', 2'\rangle$ from V (picking the other element leads to the same output, with small differences in the run of the algorithm), and executes the *Refine* block: at line 12 the relation is updated so that $R(1') = R_i(1') \cap \mathrm{pre}_a(R_i(2')) = \{0', 1'\}$.

4th iteration: As in the previous iteration, $U = \emptyset$. On the other hand, now we have $V = \{\langle a, 0', 1'\rangle\}$ since $R(0') = R_i(0') = \{0', 1', 2'\} \not\subseteq \mathrm{pre}_a(R(1')) = \{0'\}$. Therefore, the algorithm executes a *Refine* step, and at line 12 the principal $R(0')$ is refined, so that $R(0') = \{0', 1', 2'\} \cap \{0'\} = \{0'\}$.

5th iteration: As for the previous iterations, $U = \emptyset$ holds since $\sigma = \{0', 0\}$, and for every $x \in \mathbb{N} \cup \{0', 1', 2'\}$, it holds $R(x) \cap \sigma \neq \emptyset$. Moreover, we have that $V = \emptyset$, since all the transitions outgoing $0'$, $1'$, $2'$, and all the states in $\mathbb{N}$ are stable. Therefore, Algorithm 1 returns $\sigma = \{0', 0\}$, and R is defined by the following principals: $R(0') = \{0'\}$, $R(1') = \{0', 1'\}$ and $R(x) = R_i(x)$ for every other state. Observe that $|\sigma| = 2$, independently of the fixed parameter k—for an arbitrary value k, every principal $R_{\mathrm{sim}}(0'), \ldots, R_{\mathrm{sim}}(k')$ contains the state $0'$, and therefore adding $0'$ into σ at the first iteration suffices to make them all provably reachable. In fact, for this family of transition systems, Algorithm 1 explores only two reachable states 0 and $0'$ (out of infinitely many) which suffice to characterize all the reachable principals of R_{sim}.

6 Computing the Reachable Simulation Partition

Since, for finite state systems, it is possible to compute precisely the reachable blocks of the simulation partition in a naïve way by first computing the simulation partition blocks, and then filtering out the ones containing no reachable state, it is a natural question to ask whether Algorithm 1 can be modified to achieve a similar result. To answer this question, we first study in Section 6.1 the ways in which we can obtain an approximate solution for Problem 4.1. Then, we introduce, in Section 6.2, the auxiliary Algorithm 2 which we will then employ to modify Algorithm 1 into Algorithm 3 presented in Section 6.3, which, by adopting a different approximation strategy, turns the containment of equation (1.b) into an equality.

6.1 Approximating Reachable Blocks

We focus on how the set of reachable blocks of P_{sim} can be approximated. We recall that the set of reachable P_{sim} blocks is defined as $P_{\text{sim}}^{\text{post}^*(I)}$. Let $\{B_1, B_2, \dots\} = P_{\text{sim}}^{\text{post}^*(I)}$ be the reachable blocks of P_{sim}, we identify two ways in which $P_{\text{sim}}^{\text{post}^*(I)}$ can be approximated:

Approximation by *over-reachability*: the set $P_{\text{sim}}^{\text{post}^*(I)}$ is approximated as $\alpha(\{B_1, B_2, \dots\})$, where $\alpha : \wp(P_{\text{sim}}) \to \wp(P_{\text{sim}})$ (possibly) introduces some degree of over-approximation by adding new blocks to the set, i.e., $\alpha(P_{\text{sim}}^{\text{post}^*(I)}) = \{B_1, B_2, \dots, C_1, C_2, \dots\} \supseteq P_{\text{sim}}^{\text{post}^*(I)}$, where (some of) the C_i blocks might not belong to $P_{\text{sim}}^{\text{post}^*(I)}$. Observe that, in this scenario, the states contained in each block of $\alpha(P_{\text{sim}}^{\text{post}^*(I)})$ are exactly computed, i.e. $\alpha(P_{\text{sim}}^{\text{post}^*(I)}) \subseteq P_{\text{sim}}$ (meaning that each block in the approximation is an actual block of P_{sim}), and the approximation only affects *which* blocks are marked as reachable.

Approximation by *partial computation*: the set $P_{\text{sim}}^{\text{post}^*(I)}$ is approximated as $\{\beta(B_1), \beta(B_2), \dots\}$, where the content of each reachable block is approximated by an injective map $\beta : P_{\text{sim}}^{\text{post}^*(I)} \to \wp(\Sigma)$ such that $\beta(B_i) \subseteq B_i$. This kind of approximation induces a 1-1 correspondence between each B_i and the corresponding $\beta(B_i)$. In this scenario, only the content of the reachable blocks is approximated, but not the characterization of which blocks are reachable, dually to what happens in the over-reachability case above.

We observe that the notion of approximation by partial computation bears resemblance to that of *semi-stability* used by Lee and Yannakakis [1992]. In fact, as far as model minimization is concerned, computing the set $P_{\text{sim}}^{\text{post}^*(I)}$ in an exact way gives rise to the same graph structure as approximating the output via partial computation, similarly to how a semi-stable graph has the same "shape" of the reduced system, in [Lee and Yannakakis 1992, Definition 3.2].

As we have shown in Theorem 5.3, (1.b), Algorithm 1 (and the same holds for Algorithm 4 in the upcoming section) solves an approximated by *over-reachability* version of Problem 4.1 for P_{sim} blocks. Intuitively, we will now formulate a modified version of Algorithm 1 which solves the dually approximated problem for *partial computation*.

More in detail, the algorithm we propose offers strong guarantees on the degree of approximation introduced in each block, as the function β satisfies the following property: for every $B \in P_{\text{sim}}^{\text{post}^*(I)}$, it holds $B \cap \text{post}^*(I) = \beta(B) \cap \text{post}^*(I)$ (as stated in equation (2.b) of Theorem 6.1 below). This therefore means that the effects of the approximation introduced by β are limited to the unreachable states of each block, while the reachable states are computed in an exact way.

6.2 A Refined Version of Algorithm 1

The purpose of this section is to introduce a partially evaluated version of Algorithm 1, given as Algorithm 2 below, which will be used as a subroutine of Algorithm 3 in Section 6.3. The intuition underlying Algorithm 2 is that if Algorithm 1 is called with $\sigma_i = \text{post}^*(I)$, then the set U will be empty at each iteration, so that *Search* never executes.

Algorithm 2: Refined Algorithm (SymRef)

Input: A transition system $G = (\Sigma, I, L, \rightarrow)$, the set of reachable states $\mathrm{post}^*(I)$, and a relation $R_i \in \mathrm{Rel}(\Sigma)$ such that $R_{\mathrm{sim}} \subseteq R_i \subseteq R_{\mathrm{po}}$, where R_{po} is some preorder and R_{sim} is the simulation preorder induced by R_{po}.

```
1  Rel(Σ) ∋ R := R_i;
2  ℘(Σ) ∋ σ := post*(I);
3  while true do
       // Inv_1 : ∀x. R_sim(x) ⊆ R(x) ⊆ R_i(x)
       // Inv_2 : ∀x ∈ Σ. x ∈ R(x)
4      V := {⟨a, x, x'⟩ ∈ L × Σ^2 | R(x) ∩ σ ≠ ∅, x -a-> x', R(x) ⊈ pre_a(R(x'))};
5      if (V ≠ ∅) then
6          Refine :
7              choose ⟨a, x, x'⟩ ∈ V;
8              R(x) := R(x) ∩ pre_a(R(x'));
9      else
10         return ⟨R, σ⟩;
```

Based on this observation, we define Algorithm 2, also denoted by SymRef, obtained by partially evaluating Algorithm 1 under the assumption to have in input $\sigma_i = \mathrm{post}^*(I)$. Note that Algorithm 2 differs from Algorithm 1 in that we require the input relation R_i to be reflexive, but not necessarily transitive. The rationale is that Algorithm 2 will be employed as a subroutine having in input a relation R_i that sits in between a preorder R_{po} and the simulation preorder R_{sim} induced by R_{po}. More precisely, it turns out that for Algorithm 2 the invariant $\textsc{Inv} \triangleq R_{\mathrm{sim}} \subseteq R_i \subseteq R_{\mathrm{po}}$ holds. Note that R_i is reflexive, since R_{sim} is, but R_i need not be transitive. Under these assumptions on the inputs, it is easily seen that Algorithm 2 inherits correctness and termination results as given by Theorems 5.3 and 5.4.

6.3 An Algorithm for the Reachable Simulation Partition

We now introduce the key modification underlying the adapted version of Algorithm 1. Intuitively, our modification changes the notions of reachability for principals in the sets U and V, switching from that defined in (rp_1) to the one given by (rp_2). Moreover, a third guarded block is added to the **nif** statement of Algorithm 1, which computes all the one-step successors of states in σ and checks whether the whole $\mathrm{post}^*(I)$ set has been computed. Like Algorithm 1, this modified version is guaranteed to terminate on finite state systems.

Let us recall that, following the idea presented in Example 4.5, computing the whole set $\mathrm{post}^*(I)$ is, in general, unavoidable. In particular, several features of this modified algorithm suggest that it terminates less often for infinite state systems compared to Algorithm 1, because, intuitively, it relies more tightly on explicitly computing reachable states (e.g., some part of the algorithm explicitly relies on having computed the whole set $\mathrm{post}^*(I)$).

Algorithm 3: Reachable Simulation Partition Algorithm

Input: A ts $G=(\Sigma, I, L, \rightarrow)$, $R_i \in \mathrm{PreO}(\Sigma)$, and an initial finite set $I \subseteq \sigma_i \subseteq \mathrm{post}^*(I)$.

1 $\mathrm{Rel}(\Sigma) \ni R := R_i$;
2 $\wp(\Sigma) \ni \sigma := \sigma_i$;
3 $\wp(U) \ni U_{\mathrm{bad}} := \varnothing$;
4 **while true do**
  // $\textsc{Inv}_1$: $\forall x \in \Sigma.\, R_{\mathrm{sim}}(x) \subseteq R(x) \subseteq R_i(x)$
  // $\textsc{Inv}_2$: $\sigma_i \subseteq \sigma \subseteq \mathrm{post}^*(I)$
  // $\textsc{Inv}_3$: $\forall x \in \Sigma.\, x \in R(x)$
  // $\textsc{Inv}_4$: $U_{\mathrm{bad}} \subseteq U$
5 $U := \{x \in \Sigma \mid \nexists s \in \sigma.\, R(x) = R(s),\ R(x) \cap \mathrm{post}(\sigma) \neq \varnothing\}$;
6 $V := \{(a, x, x') \in L \times \Sigma^2 \mid \exists s \in \sigma.\, R(x) = R(s),\ x \xrightarrow{a} x',\ R(x) \not\subseteq \mathrm{pre}_a(R(x'))\}$;
7 **nif**
8 $(U \setminus U_{\mathrm{bad}} \neq \varnothing) \longrightarrow$ *Search* :
9 **choose** $x \in U \setminus U_{\mathrm{bad}}$;
10 $S := (R(x) \cap \mathrm{post}(\sigma)) \setminus \sigma$;
11 **if** $S \neq \varnothing$ **then**
12 **choose** $s \in S$;
13 $\sigma := \sigma \cup \{s\}$;
14 $U_{\mathrm{bad}} := \varnothing$;
15 **else**
16 $U_{\mathrm{bad}} := U_{\mathrm{bad}} \cup \{x\}$;
17 $(V \neq \varnothing) \longrightarrow$ *Refine* :
18 **choose** $(a, x, x') \in V$;
19 $S := \mathrm{pre}_a(R(x'))$;
20 $R(x) := R(x) \cap S$;
21 $U_{\mathrm{bad}} := \varnothing$;
22 $(U = U_{\mathrm{bad}} \neq \varnothing \wedge V = \varnothing) \longrightarrow$ *Expand* :
23 **if** $\mathrm{post}(\sigma) \subseteq \sigma$ **then**
  // $\sigma = \mathrm{post}^*(I)$ holds, thus Algorithm 2 is executed with input R and σ
24 **return** $\mathrm{SymRef}(R, \sigma)$;
25 **else**
26 $\sigma := \sigma \cup \mathrm{post}(\sigma)$;
27 $U_{\mathrm{bad}} := \varnothing$;
28 $(U = \varnothing \wedge V = \varnothing) \longrightarrow$ **return** $\langle R, \sigma \rangle$;

We now illustrate, in detail, the differences between Algorithm 1 and Algorithm 3. One first difference lies in the definitions of the sets U and V at lines 5 and 6. Informally, the changes made to the definitions of U and V reflect the difference between definitions (rp_1) and (rp_2) of reachable principal. Moreover, Algorithm 3 keeps track of a subset of U, denoted by U_{bad}, that contains states on which executing a *Search* iteration cannot expand σ (using the current relation R). Furthermore, we have a new *Expand* guarded block in the loop of nondeterministic choices whose role is to guarantee progress by adding new elements to σ. The *Expand* block at line 24 has a return statement invoking Algorithm 2 as a subroutine whenever σ turns out to be the whole set $\mathrm{post}^*(I)$ of reachable

states. It is worth recalling that, as shown in Example 4.5, computing the whole set $\mathrm{post}^*(I)$ of reachable states is, in general, unavoidable.

The main feature of Algorithm 3 is that if it terminates with output $\langle R, \sigma\rangle$, then it induces a partition whose blocks having a nonempty intersection with σ approximate, in a *partial computation* way, the reachable blocks of P_{sim}, splitting every reachable state consistently with P_{sim}, where *consistently* here means that two reachable states are split by P iff they are split by P_{sim} (cf. (2.b) below). Note that a block $B \in P^\sigma$ might be strictly contained in the corresponding block $P_{\mathrm{sim}}(B)$ of P_{sim}, but Algorithm 3 ensures that $(P_{\mathrm{sim}}(B) \smallsetminus B) \cap \mathrm{post}^*(I) = \varnothing$ (cf. (2.c) below). We also point out that, as a consequence of (2.b), every block of $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$ intersects σ. On the other hand, Algorithm 3 is also precise for the reachable principals of the simulation preorder R_{sim} where reachability for principals is defined according to definition (rp_2) (cf. (2.a) below).

Theorem 6.1 (**Finite Correctness of Algorithm 3**). *Let $\langle R, \sigma\rangle \in \mathrm{Rel}(\Sigma) \times \wp(\Sigma)$ be the output of Algorithm 3 on input G with $|\Sigma| \in \mathbb{N}$, $R_i \in \mathrm{PreO}(\Sigma)$, $I \subseteq \sigma_i \subseteq \mathrm{post}^*(I)$. Moreover let $P \in \mathrm{Part}(\Sigma)$ be the partition induced by R. Then:*

$$(R_{\mathrm{sim}})_{alt}^{\mathrm{post}^*(I)} = R_{alt}^{\mathrm{post}^*(I)}, \tag{2.a}$$

$$\{B \cap \mathrm{post}^*(I) \mid B \in P_{\mathrm{sim}}^{\mathrm{post}^*(I)}\} = \{B \cap \mathrm{post}^*(I) \mid B \in P^\sigma\}, \tag{2.b}$$

$$\forall x \in \Sigma.\, P(x) \in P^\sigma \Rightarrow P(x) \subseteq P_{\mathrm{sim}}(x)\ . \tag{2.c}$$

Proof. We first observe that at termination $V = \varnothing$ and one of the two following holds:

$$U = \varnothing \tag{3}$$

$$U \neq \varnothing \wedge (\mathrm{post}(\sigma) \subseteq \sigma) \tag{4}$$

corresponding to, respectively, the case where the algorithm executes line 28 or 24. Moreover, in the case $U \neq \varnothing$, we observe that $\mathrm{post}(\sigma) \subseteq \sigma$, together with $I \subseteq \sigma$, allows us to conclude that $\mathrm{post}^*(I) \subseteq \sigma$, by fixpoint definition of $\mathrm{post}^*(I)$, and Inv_2 implies $\sigma = \mathrm{post}^*(I)$. Moreover, since Algorithm 2 returns the same set of reachable states, i.e. $\mathrm{post}^*(I)$, which is given as input, we have that $\sigma = \mathrm{post}^*(I)$ holds.

We now show that the following facts on the output pair $\langle R, \sigma\rangle$ of Algorithm 3 hold:

(i) For all $x \in \Sigma$, $R_{\mathrm{sim}}(x) \subseteq R(x)$.
This holds because the *Refine* block of Algorithm 3 is always correct, so that, for all $x \in \Sigma$, $R_{\mathrm{sim}}(x) \subseteq R(x) \subseteq R_i(x)$ holds for every iteration of Algorithm 3. Moreover, correctness of Algorithm 2 ensures that line 24 preserves this property.

(ii) $x \in \mathrm{post}^*(I) \Rightarrow \exists s \in \sigma.\, R(x) = R(s)$.
We distinguish the two possible cases at termination:
($U = \varnothing$): We proceed by induction on $n \in \mathbb{N}$ such that $x \in \mathrm{post}^n(I)$. For $n = 0$, $x \in I$, so that by initialization of σ, $x \in \sigma$. For the inductive case, if $x \in \mathrm{post}^{n+1}(I)$ then there exists $y \in \mathrm{post}^n(I)$ such that $y \to x$, so that, by inductive hypothesis, $R(y) = R(s')$ for some $s' \in \sigma$. Since $y \to x$, $V = \varnothing$ implies $R(y) \subseteq \mathrm{pre}(R(x))$. Thus, since R is reflexive, $s' \in R(s') = R(y)$ holds, implying $s' \in \mathrm{pre}(R(x))$, i.e., $R(x) \cap \mathrm{post}(\sigma) \neq \varnothing$. Then, $U = \varnothing$ implies $\exists s \in \sigma.\, R(s) = R(x)$.
($U \neq \varnothing$): Since $\sigma = \mathrm{post}^*(I)$, we have that $x \in \mathrm{post}^*(I) = \sigma$, and the property trivially holds.

(iii) $(\exists s \in \sigma.\, R(s) = R(x)) \Rightarrow R(x) = R_{\mathrm{sim}}(x)$.
We distinguish two possible cases at termination:
($U \neq \varnothing$): By reflexivity, $s \in R(s) = R(x)$ implies $R(x) \cap \sigma \neq \varnothing$, and thus correctness of Algorithm 2 entails $R(x) = R_{\mathrm{sim}}(x)$.

($U = \varnothing$): Let Sim be the basic simulation algorithm, recalled as Algorithm 5 in Section 9, taking in input a relation R and computing, for finite state systems, R_{sim}.
By (i), $R_{\mathrm{sim}}(z) \subseteq R(z) \subseteq R_i(z)$, for all $z \in \Sigma$. Moreover, $\mathsf{Sim}(R) = R_{\mathrm{sim}}$ because $R_{\mathrm{sim}} = \mathsf{Sim}(R_{\mathrm{sim}}) \subseteq \mathsf{Sim}(R) \subseteq \mathsf{Sim}(R_i) = R_{\mathrm{sim}}$. Let us assume, by contradiction, that $\mathbb{S} \triangleq \{R(x) \mid x \in \Sigma, R(x) \neq R_{\mathrm{sim}}(x), \exists s \in \sigma.\, R(x) = R(s)\} \neq \varnothing$, so that every principal in $\mathbb{S}$ will be refined at some iteration of Sim on input R. Let $R(x) \in \mathbb{S}$ be the first principal in $\mathbb{S}$ to be refined by Sim at some iteration *it* whose current relation is R' with $R' \subseteq R$, so that each principal in $\mathbb{S}$ is a principal for R' since no principal in $\mathbb{S}$ was refined before this iteration *it*. Let $R'(y) \subseteq R(y)$ be the principal of R' used in this iteration *it* to refine $R(x)$, so that $x \xrightarrow{a} y$, $R'(x) = R(x) \not\subseteq \mathrm{pre}_a(R'(y))$. We have that $x \xrightarrow{a} y$, along with $\exists s \in \sigma.\, R(s) = R(x)$ and $V = \varnothing$, entails $R(x) \subseteq \mathrm{pre}_a(R(y))$. Hence, by reflexivity, $s \in R(s) = R(x) \subseteq \mathrm{pre}_a(R(y))$ implies $R(y) \cap \mathrm{post}(\sigma) \neq \varnothing$, so that $U = \varnothing$ entails $\exists s' \in \sigma.\, R(y) = R(s')$.
Now, either $R(y) \notin \mathbb{S}$, which implies $R(y) = R_{\mathrm{sim}}(y)$ and, therefore, $R(y) = R'(y)$, or $R(y) \in \mathbb{S}$, and since no principal in $\mathbb{S}$ was refined before, $R(y) = R'(y)$.
This lets us conclude that $R(x) \not\subseteq \mathrm{pre}_a(R'(y)) = \mathrm{pre}_a(R(y))$, which is a contradiction to $R(x) \subseteq \mathrm{pre}_a(R(y))$. Thus, $\mathbb{S} = \varnothing$ holds.

(iv) $(\exists s \in \sigma.\, R(s) = R(x)) \Rightarrow P(x) \cap \mathrm{post}^*(I) = P_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I)$.
We first show that

$$(\exists s \in \sigma.\, R(s) = R(x)) \Rightarrow P(x) \subseteq P_{\mathrm{sim}}(x). \tag{5}$$

Consider $y \in P(x)$. Then, by definition of P it holds $R(y) = R(x)$, and, thus, $R(y) = R(s)$. Moreover, by (iii), we get $R_{\mathrm{sim}}(y) = R(y) = R(x) = R_{\mathrm{sim}}(x)$, and thus, by definition of P_{sim}, it follows that $y \in P_{\mathrm{sim}}(x)$, meaning $P(x) \subseteq P_{\mathrm{sim}}(x)$, which proves (5). The inclusion $P(x) \cap \mathrm{post}^*(I) \subseteq P_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I)$ follows by definition from (5). On the other hand, pick some $y \in P_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I)$. Then, by definition of P_{sim}, we get $R_{\mathrm{sim}}(y) = R_{\mathrm{sim}}(x)$, and, by (ii), we get that $y \in \mathrm{post}^*(I)$ entails $R(y) = R(s')$ for some $s' \in \sigma$, while (iii) implies $R(y) = R_{\mathrm{sim}}(y) = R_{\mathrm{sim}}(x) = R(x)$. Thus by definition of P, $y \in P(x)$, and, therefore, $P_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \subseteq P(x) \cap \mathrm{post}^*(I)$.

(v) $s \in \sigma \Rightarrow P_{\mathrm{sim}}(s) = P_{\mathrm{sim}}(P(s))$.
Consider $s \in \sigma$. Then, by (5), $s \in P(s) \subseteq P_{\mathrm{sim}}(s)$, and, by definition of additive lifting, we have that $\bigcup_{x \in \{s\}} P_{\mathrm{sim}}(x) \subseteq \bigcup_{x \in P(s)} P_{\mathrm{sim}}(x) \subseteq \bigcup_{x \in P_{\mathrm{sim}}(s)} P_{\mathrm{sim}}(x)$, which, in turn, entails $P_{\mathrm{sim}}(s) \subseteq P_{\mathrm{sim}}(P(s)) \subseteq P_{\mathrm{sim}}(P_{\mathrm{sim}}(s)) = P_{\mathrm{sim}}(s)$, and, therefore, $P_{\mathrm{sim}}(s) = P_{\mathrm{sim}}(P(s))$.

(vi) $\sigma \subseteq \mathrm{post}^*(I)$.
This holds since $\sigma_i \subseteq \mathrm{post}^*(I)$ and updates at lines 13, 26 preserve this property.

Let us show (2.a). Consider $x \in \mathrm{post}^*(I)$. Then, by (ii), there exists some $s \in \sigma$ such that $R(s) = R(x)$, and, by (iii), $R(x) = R_{\mathrm{sim}}(x)$, proving that $R_{\mathrm{sim}}(x) \in \{R(y) \mid y \in \sigma\}$. On the other hand, consider a state $s \in \sigma$. Then, by (vi), we have $s \in \mathrm{post}^*(I)$. Moreover, by (iii), we get $R(s) = R_{\mathrm{sim}}(s)$, and, thus, we get $R(s) \in \{R_{\mathrm{sim}}(y) \mid y \in \mathrm{post}^*(I)\}$.

We now prove (2.b). Consider a block $P_{\mathrm{sim}}(z)$ such that $P_{\mathrm{sim}}(z) \cap \mathrm{post}^*(I) \neq \varnothing$, so that $P_{\mathrm{sim}}(z) = P_{\mathrm{sim}}(x)$ for some $x \in \mathrm{post}^*(I)$. By (ii), $R(x) = R(s)$ for some $s \in \sigma$, and, by (iii), $R_{\mathrm{sim}}(x) = R(x) = R(s) = R_{\mathrm{sim}}(s)$, meaning that $P_{\mathrm{sim}}(s) = P_{\mathrm{sim}}(x) = P_{\mathrm{sim}}(z)$. Moreover, by (iv), we get $P_{\mathrm{sim}}(s) \cap \mathrm{post}^*(I) = P(s) \cap \mathrm{post}^*(I)$, and, by reflexivity of P, we get $P_{\mathrm{sim}}(z) \cap \mathrm{post}^*(I) \in \{B \cap \mathrm{post}^*(I) \mid B \in P,\ B \cap \sigma \neq \varnothing\}$. For the other inclusion, we consider $P(z)$ such that $P(z) \cap \sigma \neq \varnothing$, so that $P(z) = P(s)$ for some $s \in \sigma$ and $R(z) = R(s)$, by definition of P. By (iv), $P(z) \cap \mathrm{post}^*(I) = P_{\mathrm{sim}}(z) \cap \mathrm{post}^*(I)$. Moreover, (vi) entails $s \in P(z) \cap \mathrm{post}^*(I)$, and, therefore, $s \in P_{\mathrm{sim}}(z) \cap \mathrm{post}^*(I)$, meaning that $P_{\mathrm{sim}}(z) \cap \mathrm{post}^*(I) \neq \varnothing$. This proves that $P(z) \cap \mathrm{post}^*(I) \in \{B \cap \mathrm{post}^*(I) \mid B \in P_{\mathrm{sim}},\ B \cap \mathrm{post}^*(I) \neq \varnothing\}$.

Finally, we prove (2.c). Consider $x \in \Sigma$ such that $P(x) \in P^\sigma$. Then, $P(x) \in P^\sigma$ entails $P(x) = P(s)$ for some $s \in \sigma$. We now observe that $P(s) \subseteq P_{\mathrm{sim}}(P(s))$ by extensivity, and, since (v) entails $P_{\mathrm{sim}}(P(s)) = P_{\mathrm{sim}}(s)$, we conclude that $P(s) \subseteq P_{\mathrm{sim}}(s)$. □

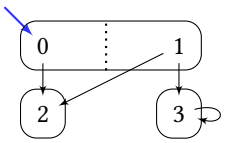
As we did for Algorithm 1, we state the correctness result in Theorem 6.1 for finite transition systems only, allowing for a more concise proof. Nevertheless, the correctness result can be extended to the infinite system case as follows.

Theorem 6.2 (**Correctness of Algorithm 3**). *Let $\langle R, \sigma\rangle \in \mathrm{Rel}(\Sigma) \times \wp(\Sigma)$ be the output of Algorithm 3 on input G with $R_i \in \mathrm{PreO}(\Sigma)$ and $I \subseteq \sigma_i \subseteq \mathrm{post}^*(I)$. Moreover, let $P \in \mathrm{Part}(\Sigma)$ be the partition induced by R. If Assumption 5.5 holds, then conditions (2.a), (2.b), and (2.c) are satisfied.*

The extension of the proof to the case of infinite-state transition systems is deferred to Section 9, as we did for the other correctness proofs.

The following example shows the role of the *Expand* iterations and of the call of Algorithm 2 at line 24 in ensuring termination.

Example 6.3. Consider the finite-state system depicted below, where the initial preorder R_i is given by $R_i(0) = R_i(1) = \{0, 1\}$, $R_i(2) = \{2\}$, and $R_i(3) = \{2, 3\}$. Boxes in the diagram gather states sharing the same principal in R_i, that is, blocks of the partition induced by R_i. The simulation preorder R_{sim} w.r.t. R_i is such that $R_{\mathrm{sim}}(0) = \{0, 1\}$, $R_{\mathrm{sim}}(1) = \{1\}$, $R_{\mathrm{sim}}(2) = \{2\}$, and $R_{\mathrm{sim}}(3) = \{3\}$, meaning that states 0 and 1 are split in P_{sim}, and, as for the previous examples, dotted lines delimit the blocks of P_{sim}. The initial state is $I = \{0\}$, and $\mathrm{post}^*(I) = \{0, 2\}$.



Executing Algorithm 3 on input R_i and $\sigma_i = I$, we reach $\sigma = \{0, 2\}$ after the first *Search* iteration. At this point, $V = \emptyset$ and $U = \{3\}$, and the state 3 will be inserted into U_{bad} after a further *Search* iteration, since $\sigma = \mathrm{post}^*(I)$ and, therefore, it cannot be expanded any further. Executing an *Expand* iteration is then unable to expand σ since $\mathrm{post}(\sigma) \subseteq \sigma$, so that the algorithm will execute the call $\mathsf{SymRef}(R, \sigma)$ of Algorithm 2 at line 24, thus executing the refinement $R(1) = \{1\}$ before returning. This refinement induces the separation of the states 0 and 1 in the partition P induced by the relation R output by the algorithm.
It is worth remarking that the subroutine $\mathsf{SymRef}(R, \sigma)$ plays an important role in this example, since it induces the split, in the partition P induced by the output relation R, of the two states 0 and 1, ensuring that the reachable state 0 is not erroneously classified as equal to 1. ◇

As aforementioned, it turns out that Algorithm 3 always terminates on finite state systems.

Theorem 6.4 (**Termination of Algorithm 3**). *Let $G = (\Sigma, I, L, \to)$ with $|\Sigma| \in \mathbb{N}$, $I \subseteq \sigma_i \subseteq \mathrm{post}^*(I)$, and $R_i \in \mathrm{PreO}(\Sigma)$. Then, Algorithm 3 terminates on input G, R_i and σ_i.*

Proof. We first observe that executing the *Refine* block is guaranteed to remove some state from at least one principal of the current relation, and since each principal has an initial finite number of elements, the overall number of *Refine* iterations is finite. Then, every *Expand* iteration either adds some new state to σ through the update at line 26, or it reaches the return statement at line 24. Hence, the number of *Expand* iterations is also finite. Moreover, every *Search* iteration will either add some new state to σ through the update at line 13, or some new state to U_{bad} through the update at line 16. Thus, since Σ is a finite set, the algorithm will always execute a finite number of *Search*

iterations which expand σ. Finally, the number of *Search* iterations expanding U_{bad} occurring in between two iterations resetting U_{bad} to $\varnothing$ is also finite, since $U_{\text{bad}} \subseteq \Sigma$ and Σ is finite, and observing that the iterations executing $U_{\text{bad}} := \varnothing$ are, in turn, either *Refine* iterations, non-terminating *Expand* iterations, or *Search* iterations expanding σ, which we have shown to be finitely many, allows us to conclude that the total number of *Search* iterations is also finite.
Therefore, since every iteration of the while-loop executes either a *Search*, *Refine* or *Expand* iteration, the algorithm terminates after a finite number of iterations. Observing that each iteration computes a finite number of operations, and, in particular, the execution of Algorithm 2 at line 24 is guaranteed to terminate by Theorem 5.4, completes the proof. □

7 2PR Triples

Symbolic approaches for simulation algorithms based on state partitions are essential and beneficial for algorithms processing infinite state systems, as shown by Henzinger et al. [1995] for the symbolic simulation algorithm on infinite graphs and, in particular, hybrid automata. Moreover, symbolic approaches are also advantageous in terms of space and time efficiency for finite state systems [Cécé 2017; Crafa et al. 2011; Ranzato 2013]. We introduce 2-Partitions-Relation triples (2PR), generalizing the partition-relation pairs used in the most efficient simulation algorithms as a symbolic representation of a relation between states [Cécé 2017; Gentilini et al. 2003; Ranzato and Tapparo 2010]. We exploit 2PRs to design a symbolic version of Algorithm 1 in Section 8. The rationale behind the need for 2PRs rather than partition-relation pairs (viz. 1PR) has more to do with enhancing the presentation and ease of understanding and less to do with limitations of 1PRs. As we will observe in the upcoming sections, the use of 2PR will improve termination, since our symbolic algorithm can terminate on many inputs where Algorithm 1 does not.

*Definition 7.1 (**2PR Triple**).* Given an (infinite) set Σ, a triple $\langle P, \tau, Q\rangle$ with $P, Q \in \mathrm{Part}(\Sigma)$ and $\tau\colon P \longrightarrow \wp(Q)$, is a *2-Partitions-Relation* (2PR) *triple*. ◇

A relation $R \in \mathrm{Rel}(\Sigma)$ induces a 2PR triple $\langle P_R, \tau_R, Q_R\rangle$ where P_R and Q_R are the partitions induced by R and R^{-1}, respectively, and the function τ_R is given by $\tau_R(B) \triangleq \{C \in Q_R \mid C \subseteq R(B)\}$. Conversely, a 2PR triple $\langle P, \tau, Q\rangle$ defines a relation $R_{\langle P,\tau,Q\rangle} \in \mathrm{Rel}(\Sigma)$ defined as $R_{\langle P,\tau,Q\rangle}(x) \triangleq \cup\tau(P(x))$, i.e., the union of the blocks in $\tau(P(x))$. In the following, $R_{\langle P,\tau,Q\rangle}$ is called the relation underlying the 2PR triple $\langle P, \tau, Q\rangle$, when no ambiguity arises. Additionally, we denote with $P_{\langle P,\tau,Q\rangle}$ the partition induced by $R_{\langle P,\tau,Q\rangle}$, and it is routine to check that $P \preceq P_{\langle P,\tau,Q\rangle}$.

We prove some properties of 2PR triples.

Property 7.2. For any $R \in \mathrm{Rel}(\Sigma)$ (thus, no assumption on R), $R = R_{\langle P_R,\tau_R,Q_R\rangle}$ holds.

Proof. We first observe that for all $x \in \Sigma$, $R(x) = R(P_R(x))$, by additive lifting of R over the elements of $P_R(x)$, and the fact that $z \in P_R(x)$ iff $R(z) = R(x)$. Moreover, P_R and Q_R are both partitions of Σ, so that, for all $x \in \Sigma$, both $x \in P_R(x)$ and $x \in Q_R(x)$. We prove the two inclusions for the property.

($\subseteq$): Let $y \in R(x)$. Then, by definition of P_R, $x \in P_R(x)$, and, similarly, $y \in Q_R(y)$. We show that $Q_R(y) \in \tau_R(P_R(x))$: in fact, for all $z \in Q_R(y)$, $x \in R^{-1}(y) = R^{-1}(z)$ holds, and since $R(x) = R(P_R(x))$, we have that $Q_R(y) \subseteq R(P_R(x))$. Therefore, $Q_R(y) \in \tau_R(P_R(x))$ holds and, as a consequence, $y \in Q_R(y) \subseteq \cup\tau_R(P_R(x)) = R_{\langle P_R,\tau_R,Q_R\rangle}(x)$.

($\supseteq$): Let $y \in R_{\langle P_R,\tau_R,Q_R\rangle}(x)$. Then, $Q_R(y) \in \tau_R(P_R(x))$, and, therefore, $Q_R(y) \subseteq R(P_R(x))$ holds. Since Q_R is a partition and $R(P_R(x)) = R(x)$, we have that $y \in Q_R(y) \subseteq R(x)$, thus completing the proof. □

It turns out that 2PR triples encoding reflexive relations can be characterized by extensivity of the map $\lambda B \in P.\, \cup\tau(B)$.

Property 7.3. Let $\langle P, \tau, Q\rangle$ be a 2PR triple over Σ. Then,

$$R_{\langle P,\tau,Q\rangle} \text{ is reflexive} \Leftrightarrow \forall B \in P.\, B \subseteq \cup\tau(B) \quad .$$

Proof. We prove the two implications.
($\Rightarrow$): Let $B \in P$ and $x \in B$. Then, by reflexivity, $x \in R_{\langle P,\tau,Q\rangle}(x) = \cup\tau(P(x)) = \cup\tau(B)$.
($\Leftarrow$): Let $x \in \Sigma$. Then, by extensivity of $\cup\tau$, it holds $x \in P(x) \subseteq \cup\tau(P(x)) = R_{\langle P,\tau,Q\rangle}(x)$. □

Finally, it turns out that 2PR triples induced by preorders enjoy the following property.

Property 7.4. Let R be a preorder and $\langle P_R, \tau_R, Q_R\rangle$ be the induced 2PR triple. Then, $P_R = Q_R$.

Proof. We observe that since R is a preorder, then R^{-1} is a preorder too, and, therefore, since $P_R = R \cap R^{-1}$, then $Q_R = R^{-1} \cap (R^{-1})^{-1}$. Observing that $R^{-1} \cap (R^{-1})^{-1} = R \cap R^{-1}$ completes the proof. □

8 A Symbolic Algorithm

We use 2PR triples to design Algorithm 4 as a refinement of Algorithm 1 representing R as a 2PR triple. Algorithm 4 is in symbolic logical form, meaning that it symbolically represents and processes state relations as 2PR triples $\langle P, \tau, Q\rangle$. The refinement process of this algorithm preserves reflexivity of the underlying relation (as in Algorithm 1) and during execution, for each state x, the set $R_{\langle P,\tau,Q\rangle}(x)$ includes the states candidate to simulate x, and the states in $P_{\langle P,\tau,Q\rangle}(x)$ are candidates to be simulation equivalent to x. As we will discuss in Section 8, 2PR triples bring benefits in terms of termination on infinite systems: intuitively, removing blocks from $\tau(B)$, for some block $B \in P$, updates the principal $P_{\langle P,\tau,Q\rangle}(b)$ for every $b \in B$ at once, and therefore enables the refinement of infinitely many principals in a single step.

Following the idea of Algorithm 1, this symbolic procedure computes a set of principals $\{\cup\tau(B) \mid \cup\tau(B) \cap \sigma \neq \emptyset\}_{B\in P}$, each of which is provably reachable. A block B is in U at line 8 if $\cup\tau(B)$ contains no provably reachable state, while it contains either an initial state or a successor of a provably reachable state. Also, the set V at line 11 contains unstable triples, i.e., $\langle a, B, C\rangle$ is in V iff $\cup\tau(B)$ is provably reachable and there exist $b \in B, c \in C$ such that $R_{\langle P,\tau,Q\rangle}(b)$ is a-unstable w.r.t. $R_{\langle P,\tau,Q\rangle}(c)$. Algorithm 4 either updates reachability for the principal of some block in U, executing a *Search* iteration, or stabilizes the pair of blocks associated to some triple in V, executing a *Refine* iteration. *Refine* iterations a-stabilize a pair of blocks (B, C) by possibly splitting B into $B \cap \mathrm{pre}_a(C)$ and $B \setminus \mathrm{pre}_a(C)$ (lines 13–15). Then, it refines the principal $\cup\tau(B \cap \mathrm{pre}_a(C))$ at lines 16–18 by first splitting blocks of Q, if they contain states occurring in different sets of principals (for the current relation $R_{\langle P,\tau,Q\rangle}$), and, successively, by removing at line 19 all the blocks not contained in $\mathrm{pre}_a(\cup\tau(C))$.

At termination, $R^{\sigma}_{\langle P,\tau,Q\rangle}$ coincides with the set of reachable principals of R_{sim} (cf. (3.a) below), while we obtain an over-approximation of $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$ (cf. (3.b) below). Similarly to Algorithm 1, the term "over-approximation" is w.r.t. the set $P_{\mathrm{sim}}^{\mathrm{post}^*(I)}$ itself, and not to the contained blocks, whose elements are computed in an exact way (that is, we are approximating by *over-reachability*, according to the definition in Section 6.1). We point out that (3.a) and (3.b) are 2PR-based reformulations of, respectively, (1.a) and (1.b).

Theorem 8.1 (**Finite Correctness of Algorithm 4**). *Let $\langle P, \tau, Q, \sigma\rangle$ be the output of Algorithm 4 on input G with $|\Sigma| \in \mathbb{N}$, $R_i \in \mathrm{PreO}(\Sigma)$, and $\sigma_i \subseteq \mathrm{post}^*(I)$. Moreover, let $P_{\langle P,\tau,Q\rangle} \in \mathrm{Part}(\Sigma)$*

Algorithm 4: 2PR-based Algorithm

Input: TS $G = (\Sigma, I, L, \rightarrow)$, $R_i \in \mathrm{PreO}(\Sigma)$, an initial finite set $\sigma_i \subseteq \mathrm{post}^*(I)$

1 $\mathrm{Part}(\Sigma) \ni P, Q \coloneqq \{y \in \Sigma \mid R_i(x) = R_i(y)\}_{x \in \Sigma}$;
2 **forall** $B \in P$ **do** $\wp(Q) \ni \tau(B) \coloneqq \{C \in Q \mid C \subseteq R_i(B)\}$;
3 $\wp(\Sigma) \ni \sigma \coloneqq \sigma_i$;
4 **while true do**
  // $\textsc{Inv}_1 : \forall x \in \Sigma.\, R_{\mathrm{sim}}(x) \subseteq R_{\langle P,\tau,Q\rangle}(x) \subseteq R_i(x)$
  // $\textsc{Inv}_2 : \sigma_i \subseteq \sigma \subseteq \mathrm{post}^*(I)$
  // $\textsc{Inv}_3 : \forall B \in P.\, B \subseteq \cup\tau(B)$
5 $U \coloneqq \{B \in P \mid \cup\tau(B) \cap \sigma = \emptyset,\ \cup\tau(B) \cap (I \cup \mathrm{post}(\sigma)) \neq \emptyset\}$;
6 $V \coloneqq \{\langle a, B, C\rangle \in L \times P^2 \mid \cup\tau(B) \cap \sigma \neq \emptyset,\ B \cap \mathrm{pre}_a(C) \neq \emptyset,\ \cup\tau(B) \not\subseteq \mathrm{pre}_a(\cup\tau(C))\}$;
7 **nif**
8 $(U \neq \emptyset) \longrightarrow$ *Search* :
9 **choose** $B \in U$, $s \in (\cup\tau(B) \cap (I \cup \mathrm{post}(\sigma)))$;
10 $\sigma \coloneqq \sigma \cup \{s\}$;
11 $(V \neq \emptyset) \longrightarrow$ *Refine* :
12 **choose** $\langle a, B, C\rangle \in V$; $S \coloneqq \mathrm{pre}_a(\cup\tau(C))$;
13 $B' \coloneqq B \cap \mathrm{pre}_a(C)$; $B'' \coloneqq B \setminus \mathrm{pre}_a(C)$;
14 $P.\mathit{replace}(B, \{B', B''\})$;
15 $\tau(B') \coloneqq \tau(B)$; $\tau(B'') \coloneqq \tau(B)$;
16 **forall** $X \in \{E \in \tau(B') \mid E \cap S \neq \emptyset, E \not\subseteq S\}$ **do**
17 $Q.\mathit{replace}(X, \{X \cap S, X \setminus S\})$;
18 **foreach** $A \in P$ **do** $\tau(A).\mathit{replace}(X, \{X \cap S, X \setminus S\})$;
19 $\tau(B') \coloneqq \{E \in \tau(B') \mid E \subseteq S\}$;
20 $(U = \emptyset \wedge V = \emptyset) \longrightarrow$ **return** $\langle P, \tau, Q, \sigma\rangle$;

be the partition induced by $R_{\langle P,\tau,Q\rangle}$. *Then:*

$$R_{\mathrm{sim}}^{\mathrm{post}^*(I)} = R_{\langle P,\tau,Q\rangle}^{\sigma}\ , \tag{3.a}$$

$$P_{\mathrm{sim}}^{\mathrm{post}^*(I)} \subseteq \{B \in P_{\langle P,\tau,Q\rangle} \mid R_{\langle P,\tau,Q\rangle}(B) \cap \sigma \neq \emptyset\}\ . \tag{3.b}$$

Proof. The proof follows the same pattern as the correctness proof of Algorithm 1, where $R_{\langle P,\tau,Q\rangle}$ plays the role of R in Theorem 5.3. We show some preliminary properties for the algorithm:

(i) At each iteration: $B \subseteq \cup\tau(B)$ for all $B \in P$.
This holds at initialization because R_i is a preorder. We show that the property is preserved at each *Refine* step. Suppose that a triple $\langle a, B, C\rangle$ is picked for refinement, and let $\langle P', \tau', Q'\rangle$ be the 2PR triple after execution of the *Refine* block. We notice that for all $X \in P$, $X \neq B \Rightarrow \cup\tau(X) = \cup\tau'(X)$, and that $\cup\tau'(B'') = \cup\tau(B)$, since the only statement removing states from $\cup\tau(X)$, for some block X, is at line 19. We now take $x \in B' = B \cap \mathrm{pre}_a(C)$ and proceed to show that $x \in \cup\tau'(B')$. Since $x \in \mathrm{pre}_a(C) \subseteq \mathrm{pre}_a(\cup\tau(C))$, we have that $x \in S$, and, therefore, $Q(x) \cap S \neq \emptyset$. We now distinguish two cases to show that $Q'(x) \subseteq S$:
($Q(x) \subseteq S$): Thus, $Q(x)$ is not split and $Q'(x) = Q(x) \subseteq S$.
($Q(x) \not\subseteq S$): Thus, $Q(x)$ is split and $Q'(x) = X \cap S \subseteq S$ by definition.

Therefore, since $Q'(x) \subseteq S$ holds, we have that $Q'(x) \in \tau(B')$ after line 19, and, therefore, we conclude $x \in \cup\tau'(B')$, completing the proof.

(ii) At each iteration: $\forall x \in \Sigma.\, R_{\text{sim}}(x) \subseteq R_{\langle P,\tau,Q\rangle}(x) \subseteq R_i(x)$.
This holds at initialization since $R_{\langle P_{R_i},\tau_{R_i},Q_{R_i}\rangle} = R_i$. Then, we show that every *Refine* step preserves the invariant. Let us consider a *Refine* iteration, the corresponding 2PR triple $\langle P,\tau,Q\rangle$ for which the property holds, the triple $\langle a,B,C\rangle \in V$ selected for refinement, and the 2PR triple $\langle P',\tau',Q'\rangle$ at the end of the *Refine* block. We first note that $x \notin B \Rightarrow \cup\tau(P(x)) = \cup\tau'(P'(x))$, and, similarly, $x \in B \setminus \text{pre}_a(C) \Rightarrow \cup\tau(P(x)) = \cup\tau'(P'(x))$, since the *Refine* step splits no block in P other than B and updates the underlying principal for states in $B \cap \text{pre}_a(C)$ only. Thus, we proceed by assuming $x \in B \cap \text{pre}_a(C)$, and show that for every $y \in \cup\tau(P(x)) \setminus \cup\tau'(P'(x))$, we have that $y \notin R_{\text{sim}}(x)$. To do so, we observe that $Q'(y) \notin \tau'(P'(x))$ implies that line 19 removed the block $Q'(y)$, thus meaning that $Q'(y) \nsubseteq S$. We now show that $y \notin S$ by distinguishing two cases:
$(Q(y) \cap S = \emptyset)$: This case entails $y \notin S$ trivially, by definition of intersection.
$(Q(y) \cap S \neq \emptyset)$: Since $Q'(y) \subseteq Q(y)$, we have that $Q(y) \nsubseteq S$, meaning that the block $Q(y)$ was split during the *Refine* step, that is, either $Q'(y) = Q(y) \cap S$ or $Q'(y) = Q(y) \setminus S$. The case $Q'(y) = Q(y) \cap S$ cannot happen since it would lead to the contradiction $Q'(y) \subseteq S$. Therefore, it must be the case that $Q'(y) = Q(y) \setminus S$. Observing that $y \in Q'(y)$, we conclude that $y \notin S$.
We now observe that, since $x \in B \cap \text{pre}_a(C)$, there exists some $x' \in C$ such that $x \xrightarrow{a} x'$. Since $y \notin S$ and $R_{\text{sim}}(x') \subseteq \cup\tau(P(x'))$, we can conclude that $y \notin \text{pre}_a(R_{\text{sim}}(x')) \subseteq S$, which, together with $x \to x'$, implies $y \notin R_{\text{sim}}(x)$, thus proving that $R_{\text{sim}}(x) \subseteq R_{\langle P',\tau',Q'\rangle}(x)$ holds. We now observe that the *Refine* step only removes blocks from τ, and therefore $\cup\tau'(P'(x)) \subseteq \cup\tau(P(x))$ holds for every $x \in \Sigma$. This proves that $R_{\langle P',\tau',Q'\rangle}(x) \subseteq R_{\langle P,\tau,Q\rangle}(x) \subseteq R_i(x)$ holds.

(iii) At termination: $x \in \text{post}^*(I) \Rightarrow R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \emptyset$.
This is proven by induction on $n \in \mathbb{N}$ such that $I \to^n x$. If $n = 0$ then $x \in I$, so that, since, by (i), $P(x) \subseteq \cup\tau(P(x))$, and $x \in P(x)$, it turns out that $\cup\tau(P(x)) \cap (I \cup \text{post}(\sigma)) \neq \emptyset$. Hence, $U = \emptyset$ implies that $\cup\tau(P(x)) \cap \sigma \neq \emptyset$. If $n > 0$ then $I \to^n x' \xrightarrow{a} x$, and, by inductive hypothesis, $\cup\tau(P(x')) \cap \sigma \neq \emptyset$. Therefore, since $V = \emptyset$, and $P(x') \cap \text{pre}_a(P(x)) \neq \emptyset$, then $\cup\tau(P(x')) \subseteq \text{pre}_a(\cup\tau(P(x)))$ must hold. Thus, $\text{pre}_a(\cup\tau(P(x))) \cap \sigma \neq \emptyset$, so that $\cup\tau(P(x)) \cap \text{post}_a(\sigma) \neq \emptyset$. Hence, $U = \emptyset$ implies $\cup\tau(P(x)) \cap \sigma \neq \emptyset$.

(iv) At termination: $\forall x \in \Sigma.\, R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \emptyset \Rightarrow R_{\langle P,\tau,Q\rangle}(x) = R_{\text{sim}}(x)$.
Let Sim be the basic simulation algorithm recalled as Algorithm 5 in Section 9. By (ii), $R_{\text{sim}} \subseteq R_{\langle P,\tau,Q\rangle} \subseteq R_i$. Thus, $\text{Sim}(R_{\langle P,\tau,Q\rangle}) = R_{\text{sim}}$ because $R_{\text{sim}} = \text{Sim}(R_{\text{sim}}) \subseteq \text{Sim}(R_{\langle P,\tau,Q\rangle}) \subseteq \text{Sim}(R_i) = R_{\text{sim}}$. Assume, by contradiction, that $\mathbb{S} \triangleq \{R_{\langle P,\tau,Q\rangle}(x) \in \wp(\Sigma) \mid R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \emptyset,\, R_{\langle P,\tau,Q\rangle}(x) \neq R_{\text{sim}}(x)\} \neq \emptyset$, so that each $R_{\langle P,\tau,Q\rangle}(x) \in \mathbb{S}$, will be refined at some iteration of $\text{Sim}(R_{\langle P,\tau,Q\rangle})$. Then, let $R_{\langle P,\tau,Q\rangle}(x) \in \mathbb{S}$ be the first principal in $\mathbb{S}$ such that $R_{\langle P,\tau,Q\rangle}(x)$ is refined by Sim at some iteration *it* whose current relation is R'. Thus, each principal $R_{\langle P,\tau,Q\rangle}(z) \in \mathbb{S}$ is such that $R_{\langle P,\tau,Q\rangle}(z) = R'(z)$, because no principal in $\mathbb{S}$ was refined by Sim before this iteration *it*. Let $R'(y) \subseteq R_{\langle P,\tau,Q\rangle}(y)$ be the principal of R' used by Sim in this iteration *it* to refine $R_{\langle P,\tau,Q\rangle}(x)$, so that $x \xrightarrow{a} y$ and $R'(x) = R_{\langle P,\tau,Q\rangle}(x) \nsubseteq \text{pre}_a(R'(y))$. By termination of Algorithm 4, it holds that $R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \emptyset$, $P(x) \cap \text{pre}(P(y)) \neq \emptyset$, so that $V = \emptyset$ entails $\cup\tau(P(x)) \subseteq \text{pre}_a(\cup\tau(P(y)))$. Hence, $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ implies $\cup\tau(P(y)) \cap \text{post}_a(\sigma) \neq \emptyset$, so that $U = \emptyset$ entails $\cup\tau(P(y)) \cap \sigma \neq \emptyset$, that is, $R_{\langle P,\tau,Q\rangle}(y) \cap \sigma \neq \emptyset$. If $R_{\langle P,\tau,Q\rangle}(y) \notin \mathbb{S}$ then $R_{\langle P,\tau,Q\rangle}(y) \cap \sigma \neq \emptyset$ implies $R_{\langle P,\tau,Q\rangle}(y) = R_{\text{sim}}(y) \subseteq R'(y) \subseteq R_{\langle P,\tau,Q\rangle}(y)$, so that $R'(y) = R_{\langle P,\tau,Q\rangle}(y)$ holds. If $R_{\langle P,\tau,Q\rangle}(y) \in \mathbb{S}$ then, since $R_{\langle P,\tau,Q\rangle}(x)$ is the first principal in $\mathbb{S}$ to be refined by Sim, we have that $R'(y) = R_{\langle P,\tau,Q\rangle}(y)$ holds. Thus, in both cases, $R'(y) =$

$R_{\langle P,\tau,Q\rangle}(y)$ must hold, so that $R_{\langle P,\tau,Q\rangle}(x) \not\subseteq \mathrm{pre}_a(R_{\langle P,\tau,Q\rangle}(y))$, which provides a contradiction to $\cup\tau(P(x)) \subseteq \mathrm{pre}_a(\cup\tau(P(y)))$. Thus, $\mathbb{S} = \varnothing$ must hold, and, in turn, $R_{\langle P,\tau,Q\rangle}(x) = R_{\mathrm{sim}}(x)$.

(v) At each iteration: $\sigma \subseteq \mathrm{post}^*(I)$.
This follows because $\sigma_i \subseteq \mathrm{post}^*(I)$, σ is updated at line 10 of Algorithm 4 by adding $s \in I \cup \mathrm{post}(\sigma)$, and by the fact that $\mathrm{post}^*(I)$ is the least fixpoint of $\lambda X.I \cup \mathrm{post}(X)$.

(vi) At termination: $P_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \varnothing \Rightarrow P_{\langle P,\tau,Q\rangle}(x) = P_{\mathrm{sim}}(x)$.
We first observe that, assuming $P_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \varnothing$, and considering $s \in P_{\langle P,\tau,Q\rangle}(x) \cap \sigma$, we find that $R_{\langle P,\tau,Q\rangle}(x) = \cup\tau(P(x)) = \cup\tau(P(s)) = R_{\langle P,\tau,Q\rangle}(s)$. Moreover, since by (i), $s \in P(s) \subseteq \cup\tau(P(s))$ then $R_{\langle P,\tau,Q\rangle}(s) \cap \sigma \neq \varnothing$. Therefore, by (iv), $R_{\mathrm{sim}}(x) = R_{\langle P,\tau,Q\rangle}(x) = R_{\langle P,\tau,Q\rangle}(s) = R_{\mathrm{sim}}(s)$. We prove the two inclusions of $P_{\langle P,\tau,Q\rangle}(x) = P_{\mathrm{sim}}(x)$ separately.
($\subseteq$): Assume $y \in P_{\langle P,\tau,Q\rangle}(x)$, then $\cup\tau(P(y)) = \cup\tau(P(x))$, and since $\cup\tau(P(x)) \cap \sigma \neq \varnothing$, then $\cup\tau(P(y)) \cap \sigma \neq \varnothing$. Finally, (iv) entails $R_{\mathrm{sim}}(y) = \cup\tau(P(y)) = \cup\tau(P(x)) = R_{\mathrm{sim}}(x)$, thus proving $y \in P_{\mathrm{sim}}(x)$.
($\supseteq$): Assume $y \in P_{\mathrm{sim}}(x)$, then $R_{\mathrm{sim}}(y) = R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(s)$. Moreover, since $s \in \sigma$, and $s \in R_{\mathrm{sim}}(s)$, we get $R_{\mathrm{sim}}(y) \cap \sigma \neq \varnothing$. Thus, since, by (ii), $R_{\mathrm{sim}}(y) \subseteq R_{\langle P,\tau,Q\rangle}(y)$, we have that $R_{\langle P,\tau,Q\rangle}(y) \cap \sigma \neq \varnothing$. Therefore, (iv) entails $R_{\langle P,\tau,Q\rangle}(y) = R_{\mathrm{sim}}(y)$, and therefore $R_{\langle P,\tau,Q\rangle}(y) = R_{\mathrm{sim}}(y) = R_{\mathrm{sim}}(x) = R_{\langle P,\tau,Q\rangle}(x)$ holds, proving that $\cup\tau(P(y)) = \cup\tau(P(x))$, so that $y \in P_{\langle P,\tau,Q\rangle}(x)$.

Let us now show (3.a).

($\subseteq$) Let $x \in \Sigma$ such that $R_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \varnothing$. Thus, there exists $s \in \mathrm{post}^*(I)$ such that $s \in R_{\mathrm{sim}}(x)$. By (iii), $R_{\langle P,\tau,Q\rangle}(s) \cap \sigma \neq \varnothing$. By (iv), $R_{\langle P,\tau,Q\rangle}(s) = R_{\mathrm{sim}}(s)$. Moreover, $s \in R_{\mathrm{sim}}(x)$ implies $R_{\mathrm{sim}}(s) \subseteq R_{\mathrm{sim}}(R_{\mathrm{sim}}(x)) = R_{\mathrm{sim}}(x)$, and since, by (ii), $R_{\mathrm{sim}}(x) \subseteq R_{\langle P,\tau,Q\rangle}(x)$, we obtain $R_{\mathrm{sim}}(s) \subseteq R_{\langle P,\tau,Q\rangle}(x)$. Thus, $R_{\langle P,\tau,Q\rangle}(s) \subseteq R_{\langle P,\tau,Q\rangle}(x)$ holds, so that $R_{\langle P,\tau,Q\rangle}(s) \cap \sigma \neq \varnothing$ entails $R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \varnothing$, and in turn, by (iv), $R_{\langle P,\tau,Q\rangle}(x) = R_{\mathrm{sim}}(x)$.

($\supseteq$) Let $x \in \Sigma$ such that $R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \varnothing$. By (iv), $R_{\langle P,\tau,Q\rangle}(x) = R_{\mathrm{sim}}(x)$, so that $R_{\mathrm{sim}}(x) \cap \sigma \neq \varnothing$. By (v), $R_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \varnothing$.

Let us now prove (3.b). Let $x \in \Sigma$ such that $P_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \varnothing$. Since $P_{\mathrm{sim}}(x) \subseteq R_{\mathrm{sim}}(x)$, we have that $R_{\mathrm{sim}}(x) \cap \mathrm{post}^*(I) \neq \varnothing$. Let $y \in P_{\mathrm{sim}}(x)$. Observe that $P_{\mathrm{sim}}(x) = \{z \in \Sigma \mid R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(z)\}$. Thus, $R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(y)$, so that $R_{\mathrm{sim}}(y) \cap \mathrm{post}^*(I) \neq \varnothing$. Thus, by (3.a), $R_{\mathrm{sim}}(y) = R_{\langle P,\tau,Q\rangle}(y)$ and $R_{\langle P,\tau,Q\rangle}(y) \cap \sigma \neq \varnothing$. In particular, $R_{\langle P,\tau,Q\rangle}(x) = R_{\mathrm{sim}}(x) = R_{\mathrm{sim}}(y) = R_{\langle P,\tau,Q\rangle}(y)$.

Therefore, $P_{\mathrm{sim}}(x) \subseteq \{y \in \Sigma \mid \cup\tau(P(x)) = \cup\tau(P(y))\} = P_{\langle P,\tau,Q\rangle}(x)$. On the other hand, if $z \in P_{\langle P,\tau,Q\rangle}(x)$ then $R_{\langle P,\tau,Q\rangle}(z) = \cup\tau(P(z)) = \cup\tau(P(x)) = R_{\langle P,\tau,Q\rangle}(x)$, so that $\cup\tau(P(x)) \cap \sigma \neq \varnothing$ implies $R_{\langle P,\tau,Q\rangle}(z) \cap \sigma \neq \varnothing$. Thus, by (3.a), $R_{\mathrm{sim}}(z) = R_{\langle P,\tau,Q\rangle}(z)$. Hence, $R_{\mathrm{sim}}(z) = R_{\langle P,\tau,Q\rangle}(z) = R_{\langle P,\tau,Q\rangle}(x) = R_{\mathrm{sim}}(x)$, thus proving that $P_{\langle P,\tau,Q\rangle}(x) \subseteq P_{\mathrm{sim}}(x)$, and therefore $P_{\langle P,\tau,Q\rangle}(x) = P_{\mathrm{sim}}(x)$ holds. Finally, since $R_{\langle P,\tau,Q\rangle}(x) \cap \sigma \neq \varnothing$, then $x \in P(x) \subseteq P_{\langle P,\tau,Q\rangle}(x) = P_{\mathrm{sim}}(x)$ holds (since $P_{\langle P,\tau,Q\rangle}$ is coarser than P). Thus, we obtain $R_{\langle P,\tau,Q\rangle}(P_{\mathrm{sim}}(x)) \cap \sigma \neq \varnothing$, and consequently (3.b) holds. □

As we did for Algorithm 1 and Theorem 5.3, we extend the correctness result of Theorem 8.1 to infinite state transition systems as follows:

Theorem 8.2 (**Correctness of Algorithm 4**). *Let $\langle P, \tau, Q, \sigma\rangle$ be the output of Algorithm 4 on input G, $R_i \in \mathrm{PreO}(\Sigma)$, and $\sigma_i \subseteq \mathrm{post}^*(I)$. Moreover, let $P_{\langle P,\tau,Q\rangle} \in \mathrm{Part}(\Sigma)$ be the partition induced by $R_{\langle P,\tau,Q\rangle}$. If Assumption 5.5 holds, then conditions (3.a) and (3.b) are satisfied.*

The details on how the proof of Theorem 8.1 can be extended to infinite state inputs are given in Section 9.

8.1 Implementations based on Region Algebras

We now illustrate a suitable representation for blocks of states which is needed to make the algorithm effective. What we need is a so-called *region algebra*, as described by Henzinger et al. [2005], consisting of a set $\mathcal{R}$ of regions, endowed with the operations listed below, and an extension (or interpretation) function $\lceil\cdot\rceil\colon \mathcal{R} \to \wp(\Sigma)$ mapping each region onto the states it contains. Encoding the input R_i as a 2PR triple over regions consistently with lines 1–2 of Algorithm 4 allows us to delineate, by observing the algorithm's structure, the following requirements for our region algebra:

(1) Given $s \in \Sigma$ (or, at least, $s \in \mathrm{post}^*(I)$), and $r \in \mathcal{R}$, decide if $\mathrm{post}(s) \cap \lceil r \rceil \neq \emptyset$ holds and return an element in the intersection unless it is empty;
(2) For a region r and $a \in L$, compute a region $\mathrm{pre}_a(r)$ such that $\lceil \mathrm{pre}_a(r) \rceil = \mathrm{pre}_a(\lceil r \rceil)$;
(3) For a pair of regions r, r', compute a region $r \cap r'$ such that $\lceil r \cap r' \rceil = \lceil r \rceil \cap \lceil r' \rceil$, and a region $r \setminus r'$ such that $\lceil r \setminus r' \rceil = \lceil r \rceil \setminus \lceil r' \rceil$;
(4) For a region r, decide if $\lceil r \rceil = \emptyset$ holds.

Typical examples of regions exploit symbolic constraints such as logical formulas and constraints, such as $1 \leq x \leq 9 \wedge x' = x + 1$. Once such a region algebra is defined, the input to Algorithm 4 will be: the region algebra (composed of the operators listed in points (1)–(4) above), the initial set σ_i, and a 2PR triple initialized by lines 1–2 of Algorithm 4 and encoding the initial preorder. The logical structure of the algorithm is then implemented by using the region algebra operators in input.

8.2 Termination and Complexity of Algorithm 4

We provide two conditional termination results for Algorithm 4, together with complexity bounds on the total number of its iterations. These results rely on progression guarantees and on the fact that the partitions P and Q are coarser than P_{bis} throughout the execution. Our first conditional termination result is reminiscent of [Lee and Yannakakis 1992, Th. 3.1 and the following paragraph therein] (note that the results by Lee and Yannakakis [1992] come with no proof sketches or formal proofs), and applies when the bisimulation partition P_{bis} is finite. Under that hypothesis, Algorithm 4 carries out a total number of iterations which is at most quadratic in the size of P_{bis}.

We first introduce some preliminary results which are key to deriving our termination guarantees.

Lemma 8.3 (**Refine Progression of Algorithm 4**). *Assume $\langle P, \tau, Q\rangle$ and $\langle P', \tau', Q'\rangle$ be the 2PR triples, respectively, before and after executing a Refine iteration in Algorithm 4. Then, $R_{\langle P',\tau',Q'\rangle} \subsetneq R_{\langle P,\tau,Q\rangle}$ holds.*

Proof. Let us define $R \triangleq R_{\langle P,\tau,Q\rangle}$ and $R' \triangleq R_{\langle P',\tau',Q'\rangle}$. Moreover, let $\langle a, B, C\rangle$ be the element of V picked for refinement during the *Refine* iteration. By definition of V, $\cup\tau(B) \setminus \mathrm{pre}_a(\cup\tau(C)) \neq \emptyset$. We observe that by definition of V, $B \cap \mathrm{pre}_a(C) \neq \emptyset$, so that B' is nonempty at line 13. Let $y \in \cup\tau(B) \setminus \mathrm{pre}_a(\cup\tau(C))$, so that, by definition, $y \in R(b)$. Now, we observe that $Q(y) \in \tau(B)$, and, since $y \in Q(y)$, we have that $Q \not\subseteq \mathrm{pre}_a(\cup\tau(C)) = S$. Finally, since Q' forms a partition, $y \in Q'(y)$, so that $Q'(y) \not\subseteq S$ at line 19. This allows us to conclude that $Q'(y) \notin \tau'(B')$. Thus, for $b \in B'$, we conclude that $y \in R(b)$, and $y \notin R'(b)$, thus proving that the underlying relation has been updated. □

Lemma 8.4. *Let G, $R_i \in \mathrm{PreO}(\Sigma)$, and $\sigma_i \subseteq \mathrm{post}^*(I)$ be the input of Algorithm 4. Moreover, let $P_{\mathrm{bis}} \in \mathrm{Part}(\Sigma)$ be the bisimulation partition w.r.t. the partition P_i induced by R_i. Then, at every iteration, $P_{\mathrm{bis}} \preceq P$ and $P_{\mathrm{bis}} \preceq Q$ hold.*

Proof. The proof is by induction over the iterations. The base case is trivial since at initialization $P = Q = P_i$, and, by definition, $P_{\mathrm{bis}} \subseteq P_i$. We now observe that iterations executing a *Search* step do

not update the 2PR triple, and therefore it is enough to show that the *Refine* iterations preserve the invariant.

Assume that, at the beginning of a *Refine* iteration, $P_{\text{bis}} \subseteq P$ and $P_{\text{bis}} \subseteq Q$. Let P' and Q' be, resp., the two partitions computed at the end of this iteration, and $\langle a, B, C\rangle$ be the triple selected for refinement.

We first consider the case $P \neq P'$ where, therefore, B has been split. We proceed by contradiction: suppose that there exist two states $x \in B \cap \text{pre}_a(C)$ and $y \in B \setminus \text{pre}_a(C)$ which have been separated by a split in the current iteration, and such that $P_{\text{bis}}(x) = P_{\text{bis}}(y)$. Now, since $x \in \text{pre}_a(C)$, there exists $x \xrightarrow{a} x' \in C$, and, by definition of bisimulation, there exists y' such that $y \xrightarrow{a} y'$ and $P_{\text{bis}}(x') = P_{\text{bis}}(y')$. Since, by inductive hypothesis, $P_{\text{bis}} \subseteq P$, then $P_{\text{bis}}(x') \subseteq P(x') = C$, and, therefore, $y' \in P_{\text{bis}}(y') = P_{\text{bis}}(x') \subseteq C$ implies $y' \in C$. This allows us to conclude $y \in \text{pre}_a(C)$, which gives a contradiction to $y \in B \setminus \text{pre}_a(C)$.

We consider the remaining case $Q \neq Q'$, and assume, by contradiction, that two states x, y such that $P_{\text{bis}}(x) = P_{\text{bis}}(y)$ have been separated by a split during the current iteration. Let $X \triangleq Q(x)$ be the block of Q containing x (and y) at the beginning of the iteration. Borrowing the notation from line 12, we recall that $S = \text{pre}_a(\cup\tau(C))$, and assume, w.l.o.g., that $x \in X \cap S$ and $y \in X \setminus S$ (the symmetric case is identical). Therefore, there exists $x \xrightarrow{a} x' \in \cup\tau(C)$, and, by definition of bisimulation, we obtain that there exists y' such that $y \xrightarrow{a} y'$ and $P_{\text{bis}}(x') = P_{\text{bis}}(y')$. Since, by inductive hypothesis, $P_{\text{bis}} \subseteq Q$, then $P_{\text{bis}}(x') \subseteq Q(x') \subseteq \cup\tau(C)$. Therefore, $y' \in P_{\text{bis}}(y') = P_{\text{bis}}(x') \subseteq \cup\tau(C)$, this implies $y \in \text{pre}_a(\cup\tau(C)) = S$, thus giving a contradiction to $y \in X \setminus S$. □

Lemma 8.5. *For any Search iteration of Algorithm 4, let σ and s be, respectively, the set σ at the start of the iteration and the state s selected at line 9. Then, $P_{\text{bis}}(s) \cap \sigma = \varnothing$ holds.*

Proof. Consider a *Search* iteration and let (B, s) be, resp., the block and state selected at line 9. Suppose, by contradiction, that $P_{\text{bis}}(s) \cap \sigma \neq \varnothing$. Theorem 8.4 shows that Q is coarser than P_{bis}, hence $P_{\text{bis}}(s) \subseteq Q(s)$, and, in turn, $Q(s) \cap \sigma \neq \varnothing$. By definition, $s \in \cup\tau(B)$, hence we have that $Q(s) \in \tau(B)$, and, in turn, that $\cup\tau(B) \cap \sigma \neq \varnothing$, which contradicts the definition of U at line 5. □

Theorem 8.6 (**Termination of Algorithm 4**). *Let $G, R_i \in \text{PreO}(\Sigma)$ and $\sigma_i \subseteq \text{post}^*(I)$ be the input of Algorithm 4. Moreover, let $P_{\text{bis}} \in \text{Part}(\Sigma)$ be the bisimulation partition w.r.t. the partition P_i induced by R_i. If P_{bis} has finitely many blocks then the number of iterations of Algorithm 4 is $O(|P_{\text{bis}}|^2)$.*

Proof. It follows from Theorem 8.4 that the partitions P and Q are finite at each iteration, in particular P_i contains fewer blocks than P_{bis}, and at each iteration and for every block B, the set $\tau(B) \subseteq Q$ is finite, too. Moreover, Theorem 8.4 shows that each block of P or Q coincides with the union of some blocks in P_{bis}. We capture this notion by a function $f : Q \to \wp(P_{\text{bis}})$ defined by $f : B \mapsto \{X \in P_{\text{bis}} \mid X \subseteq B\}$. It turns out that $B = \cup f(B)$, since $P_{\text{bis}} \preceq Q$.

We now show that Algorithm 4 always executes a finite number of *Refine* iterations. Lemma 8.3 ensures progression at each *Refine* iteration, meaning that, at each iteration, at least one principal is refined. We observe that, at each iteration and for each block $B \in P$, the principal $\cup\tau(B)$ coincides with a subset of P_{bis}, More precisely, $\cup\tau(B) = \cup\{\cup\{f(X) \mid X \in \tau(B)\}\}$. Therefore, let $\cup\tau'(B')$ be the principal obtained from $\cup\tau(B)$ by executing the *Refine* block. It holds that $\cup\tau'(B') \subseteq \cup\tau(B)$, which implies $\{f(X) \mid X \in \tau'(B')\} \subsetneq \{f(X) \mid X \in \tau(B)\}$. This shows that consecutive refinements of a principal induce a strictly decreasing chain over $\wp(P_{\text{bis}})$. Since P_{bis} is finite, we conclude that each block's principal can be refined at most $|P_{\text{bis}}| - 1$ times. Hence, let n be the total number of *Refine* iterations executed, we have that n is finite and $n \leq |P_{\text{bis}}|(|P_{\text{bis}}| - 1)$, since P contains fewer blocks than P_{bis}.

Then, we provide a bound to the overall number of *Search* iterations. We observe, by Lemma 8.5, that whenever a state s is picked at line 9, we have that $P_{\text{bis}}(s) \cap \sigma = \emptyset$ holds. Since s is added to σ at line 10, this implies that the number of P_{bis} blocks intersecting σ is strictly increasing at each *Search* iteration. Therefore, we conclude that no more than $|P_{\text{bis}}|$ *Search* iterations can occur. Hence, the total number of iterations is bounded by $|P_{\text{bis}}|(|P_{\text{bis}}| - 1) + |P_{\text{bis}}|$ which gives the $O(|P_{\text{bis}}|^2)$ bound on the number of iterations. □

A second termination result holds under a local finiteness condition: Algorithm 4 terminates if the number of blocks of P_{bis} contained in $R_i(\text{post}^*(I))$ is finite. This result comes with a quadratic bound on the number of iterations.

Theorem 8.7 (**Alternative Termination of Algorithm 4**). *Let G, $R_i \in \text{Part}(\Sigma)$ and $\sigma_i \subseteq \text{post}^*(I)$ be the input of Algorithm 4 (note that R_i is a partition). Let $P_{\text{bis}} \in \text{Part}(\Sigma)$ be the bisimulation partition w.r.t. R_i, and let $T \triangleq \{Y \in P_{\text{bis}} \mid R_i(Y) \cap \text{post}^*(I) \neq \emptyset\}$ $(= \{Y \in P_{\text{bis}} \mid Y \subseteq R_i(\text{post}^*(I))\})$. If T is finite then the number of iterations of Algorithm 4 is in $O(|T|^2)$.*

Proof. The intuition behind the proof is that during execution, block splitting in P or Q, only occurs for blocks X such that $X \subseteq \cup T$. Then, Theorem 8.4 shows that, since both P and Q are coarser than P_{bis}, only a finite number of block splits will occur, since $|T| \in \mathbb{N}$. The remaining part of the follows the same structure as in Theorem 8.6.

Observe that at any iteration, R_i is coarser than P and Q, which are themselves coarser than P_{bis}, by Theorem 8.4. Formally, $P_{\text{bis}} \preceq P, Q \preceq R_i$. As a consequence, we observe that the blocks in T cover the same set of states as the reachable blocks of R_i, that is, $\cup T = \cup\{Y \in R_i \mid Y \cap \text{post}^*(I) \neq \emptyset\}$. We first show that, at any iteration and for any block $X \in P \cup Q$, it holds:

$$R_i(X) \cap \text{post}^*(I) \neq \emptyset \Rightarrow X \subseteq \cup T\,. \tag{6}$$

In fact, for an arbitrary $x \in X$, we have that $P_{\text{bis}}(x) \subseteq X \subseteq R_i(X)$, and since all states in $R_i(X)$, and therefore in $P_{\text{bis}}(x)$, share the same principal in R_i (since R_i is the initial partition), then $R_i(P_{\text{bis}}(x)) = R_i(X)$ holds. Therefore, $R_i(P_{\text{bis}}(x)) \cap \text{post}^*(I) \neq \emptyset$, proving $x \in \cup T$, and $X \subseteq \cup T$.

Suppose that at some *Refine* iteration a triple $\langle a, B, C\rangle$ is picked. If a block $X \in P$ is split, then $B = X$ since *Refine* iterations can only split one block in P (that is, B). By definition of V (line 6), we get $\cup\tau(B) \cap \sigma \neq \emptyset$, and, by the invariants $\cup\tau(B) \subseteq R_i(B)$ and $\sigma \subseteq \text{post}^*(I)$, we have that $R_i(B) \cap \text{post}^*(I) \neq \emptyset$. Thus, equation (6) guarantees $B \subseteq \cup T$.

We now define the function f as in the proof of Theorem 8.6 and show that Algorithm 4 carries out finitely many *Refine* iterations.

We observe that, at initialization and for each block $C \in R_i$, it holds $C = R_i(C)$, since R_i is a partition. Moreover, since $P_{\text{bis}} \preceq Q$, we have that, at each iteration, the principal $\cup\tau(C)$ for some $C \in P$ coincides with a subset of P_{bis}. Formally, $\cup\tau(C) = \cup\{\cup\{f(X) \mid X \in \tau(C)\}\}$.

As a consequence, at initialization it holds, for any block $C' \in R_i^{\cup T}$, that $\cup\{\cup\{f(X) \mid X \in \tau(C')\}\} = R_i(C') = C'$, since R_i is a partition and principals and blocks coincide. Therefore, we have that $\cup\{f(X) \mid X \in \tau(C')\} = f(C')$, that is, the P_{bis} blocks contained in C' and in its principal are the same. Finally, since $C' \subseteq \cup T$ (because C' is a reachable block in R_i and is therefore covered by $\cup T$), it holds that $|f(C')| \leq |T|$, which entails $|\cup\{f(X) \mid X \in \tau(C')\}| \leq |T|$. We have thus shown that the principal w.r.t. R_i of any block in $R_i^{\cup T}$ coincides with a subset of P_{bis} whose size is bounded by $|T|$.

Let, at each iteration, $\langle P, \tau, Q\rangle$ be the current 2PR triple, and consider the 2PR triple given by $\langle P_{\text{bis}}, \rho, P_{\text{bis}}\rangle$ such that $\rho(Y) = \cup\{f(Y') \mid Y' \in \tau(P(Y))\}$. Using f and $P_{\text{bis}} \preceq P, Q$, it is routine to check that $R_{\langle P,\tau,Q\rangle} = R_{\langle P_{\text{bis}},\rho,P_{\text{bis}}\rangle}$.

Therefore, take any *Refine* iteration, and let the two 2PR triples $\langle P_{\text{in}}, \tau_{\text{in}}, Q_{\text{in}}\rangle$ and $\langle P_{\text{out}}, \tau_{\text{out}}, Q_{\text{out}}\rangle$ be, respectively, the 2PR triple at the beginning and at the end of the *Refine* block. Moreover, let $\langle P_{\text{bis}}, \rho_{\text{in}}, P_{\text{bis}}\rangle$ and $\langle P_{\text{bis}}, \rho_{\text{out}}, P_{\text{bis}}\rangle$ be the corresponding 2PR triples defined as described above.

Lemma 8.3 shows progress at each *Refine* iteration, hence we conclude that $R_{\langle P_{\text{out}},\tau_{\text{out}},Q_{\text{out}}\rangle} \subsetneq R_{\langle P_{\text{in}},\tau_{\text{in}},Q_{\text{in}}\rangle}$. Equivalently, $R_{\langle P_{\text{bis}},\rho_{\text{out}},P_{\text{bis}}\rangle} \subsetneq R_{\langle P_{\text{bis}},\rho_{\text{in}},P_{\text{bis}}\rangle}$.

Additionally, since we have shown that at every *Refine* iteration the chosen block $B \in P$ is such that $B \subseteq \cup T$, then we conclude that the algorithm refines the principal w.r.t. $R_{\langle P_{\text{bis}},\rho_{\text{in}},P_{\text{bis}}\rangle}$ for some block covered by $\cup T$, i.e., that there exists some block $B' \in T$ such that $\rho_{\text{out}}(B') \subsetneq \rho_{\text{in}}(B')$ (this holds since both sets are in $\wp(P_{\text{bis}})$ and $\cup\rho_{\text{out}}(B') \subsetneq \cup\rho_{\text{in}}(B')$).

Since we have shown that the principal w.r.t. R_i of any block in $R_i^{\cup T}$ coincides with a subset of P_{bis} of size bounded by $|T|$, then at initialization $|\rho(B)| \leq |T|$ for all $B \in T$.

Finally, since at each *Refine* iteration at least one of the principals in $\{\rho(B) \mid B \in T\}$ is refined by removing some of its elements, we conclude that no more than $|T| - 1$ refinements can occur to the principal of each block in T. This allows us to conclude that the total number of *Refine* iterations is bounded by $|T|(|T| - 1)$.

We now show that each time a state s is inserted in σ at line 10, it holds that $P_{\text{bis}}(s) \in T$. In fact, it holds that $s \in \cup\tau(B) \subseteq R_i(B)$, and since $s \in \text{post}^*(I)$, then $R_i(B) \cap \text{post}^*(I) \neq \emptyset$ holds. Therefore, it follows that $s \in \cup T$, implying $P_{\text{bis}}(s) \in T$. Moreover, Lemma 8.5 shows that each time a *Search* iteration is executed, the number of blocks in T intersecting σ strictly increases. This entails that no more than $|T|$ *Search* iterations can occur. Finally, the total number of iterations is bounded by $|T|(|T| - 1) + |T|$, which provides the $O(|T|^2)$ bound on the number of iterations. □

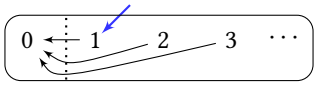
We observe that termination of Algorithm 4 can be inferred by taking into account (the number of) *Refine* iterations only, i.e., for proving termination of Algorithm 4 *Search* iterations can be neglected.

Proposition 8.8. *If the number N of Refine iterations carried out during a run of Algorithm 4 is finite, then so is the number of Search iterations. Moreover, if invariantly $|U| \leq m$ (at line 5) during a run, then that run carries out no more than $m(N + 1)$ Search iterations.*

Proof. Let U_k be the set U of line 5 at the k-th iteration. For each $k \in \mathbb{N}$, observe that, by definition, $U_k \subseteq P$, so that U_k is finite as P is initially finite and finitely many iterations (and therefore splits) have been executed. Moreover, if the k-th iteration executes a *Search* step, then $U_{k+1} \subsetneq U_k$ holds. This follows since *Search* iterations do not update the 2PR triple, and line 10 updates the set σ so as to ensure that the selected block $B \in U_k$ is such that $B \notin U_{k+1}$. Therefore, since U is strictly decreasing in consecutive *Search* iterations, Algorithm 4 does not execute consecutive *Search* iterations forever. In particular, since $|U_k| \leq m$ at each iteration, no more than m consecutive *Search* iterations take place. Hence, since no more than m *Search* iterations take place after each of the N *Refine* iterations, and, moreover, no more than m *Search* iteration take place before the first *Refine* iteration, it turns out that no more than $m(N + 1)$ *Search* iterations take place overall. □

It follows from Theorem 8.6 that Algorithm 4 terminates on all finite state systems as does Algorithm 1. However, these two termination results for Algorithm 4 are indeed stronger and extend to many infinite state systems as well. The use of 2PR triples as a representation structure brings significant benefits in terms of termination on infinite systems, and, in fact, we observe that the use of 2PR triples in Algorithm 4 significantly improves termination as there exist many inputs on which Algorithm 1 does not terminate, while Algorithm 4 does. Below we show one simple example which illustrates this phenomenon by comparing Algorithm 1 and Algorithm 4.

Example 8.9. Consider the infinite-state transition system depicted below, with $R_i = \mathbb{N} \times \mathbb{N}$. The simulation preorder R_{sim}, therefore, is such that $R_{\text{sim}}(0) = \mathbb{N}$, and $R_{\text{sim}}(n) = \mathbb{N} \setminus \{0\}$ for $n \geq 1$, meaning that $P_{\text{sim}} = \{\{0\}, \mathbb{N} \setminus \{0\}\}$. The single block induced by R_i is represented as box, and the dotted line delimits the blocks of P_{sim}.



During execution of Algorithm 1 on input R_i and $\sigma_i = I = \{1\}$, the set U is empty, since the state 1 belongs to every principal. On the other hand, the set V contains a triple $\langle a, i, 0\rangle$ for every $i \in \mathbb{N} \setminus \{0\}$. Executing a *Refine* iteration refines exactly one principal: if $\langle a, i, 0\rangle \in V$ is selected, then $R(i)$ is updated to $\mathbb{N} \setminus \{0\}$. It is easily seen that Algorithm 1 never terminates because V is never emptied, and at each iteration exactly one of them will be removed from V. ◇

On the other hand, Algorithm 4 is able to refine infinitely many principals of the underlying relation in a single *Refine* step. To illustrate this, we consider the input from Example 8.9 and show that Algorithm 4 converges in a few iterations.

Example 8.10. Consider the input of Example 8.9. The initial 2PR is given by $P = Q = \{\mathbb{N}\}$ and $\tau(\mathbb{N}) = \{\mathbb{N}\}$. Moreover, $U = \varnothing$ and V is $\{\langle a, \mathbb{N}, \mathbb{N}\rangle\}$. Executing one iteration refines both P, Q and τ, so that $P = Q = \{\{0\}, \mathbb{N} \setminus \{0\}\}$, $\tau(\{0\}) = Q$, and $\tau(\mathbb{N} \setminus \{0\}) = \{\mathbb{N} \setminus \{0\}\}$. At this point, the underlying relation is such that $R_{\langle P,\tau,Q\rangle} = R_{\text{sim}}$, and Algorithm 4 terminates. ◇

8.3 2PR Behaviour in Algorithm 4

It is worth remarking that Algorithm 4 initializes the partitions P and Q of the 2PR triple in such a way that P keeps track of states having the same principal for R_i, and Q keeps track of states occurring in the same set of principals for R_i. The idea is to start with P and Q as coarse as possible. On the opposite end, P and Q can be initialized with $P = Q$ such that each of their blocks is a singleton. Algorithm 4 with that initialization remains correct and is not much different from Algorithm 1 since the relation R of Algorithm 1 and τ of Algorithm 4 are essentially the same.

Coming back to the goal of having P and Q as coarse as possible, we show that when Algorithm 4 executes, each refinement of P (or Q) can be attributed to a change among states having the same principal for the current $R_{\langle P,\tau,Q\rangle}$ (or a change among states occurring in the same set of principals for the current $R_{\langle P,\tau,Q\rangle}$). This is formally stated as follows.

Property 8.11. Let $\langle P, \tau, Q, \sigma\rangle$ be the output of Algorithm 4 with input G, $R_i \in \text{PreO}(\Sigma)$ and $\sigma_i \subseteq \text{post}^*(I)$, and assume that the execution terminates after t iterations. Moreover, let $x, y \in \Sigma$ be two initially equivalent states (i.e., $R_i(x) = R_i(y)$), and $\langle P_j, \tau_j, Q_j\rangle$ be the 2PR triple after $j = 0, \ldots, t$ iterations of the algorithm, and $R_j = R_{\langle P_j,\tau_j,Q_j\rangle}$ be the corresponding underlying relation. Then:

$$P(x) \neq P(y) \Rightarrow R_j(x) \neq R_j(y) \quad \text{for some } j \in [0, t] \ , \tag{2PR.a}$$

while a dual property for Q holds:

$$Q(x) \neq Q(y) \Rightarrow R_j^{-1}(x) \neq R_j^{-1}(y) \quad \text{for some } j \in [0, t] \ . \tag{2PR.b}$$

Proof. We show that the following implications hold for any $k \in [0, t]$:

$$\big(\forall j \in [0, k].\, R_j(x) = R_j(y)\big) \Rightarrow P_k(x) = P_k(y)$$
$$\big(\forall j \in [0, t].\, R_j^{-1}(x) = R_j^{-1}(y)\big) \Rightarrow Q_k(x) = Q_k(y)$$

We proceed by induction on k. Assume $k = 0$, then $R_i(x) = R_i(y)$ implies $P_{R_i}(x) = P_{R_i}(y)$ by definition of 2PR triples, and since $P_{R_i} = P_0$, then $P_k(x) = P_k(y)$ holds. Moreover, Property 7.4

ensures that $P_0 = Q_0$, so that $Q_k(x) = Q_k(y)$ holds. For the inductive case $k+1$, assume that $P_k(x) = P_k(y)$ and $Q_k(x) = Q_k(y)$. Suppose, by contradiction, that $P_{k+1}(x) \neq P_{k+1}(y)$. Then, this implies that the $k+1$-th iteration was a *Refine* one in which a triple $\langle a, P_k(x), C\rangle$ was selected. Assume that $x \in P_k(x) \cap \mathrm{pre}_a(C)$ and $y \in P_k(x) \setminus \mathrm{pre}_a(C)$, so that, by (8.3), $R_{k+1}(x) \subseteq R_k(x)$. Now since the *Refine* block does not update the underlying principal for states in $P_k(x) \setminus \mathrm{pre}_a(C)$, we get $R_k(y) = R_{k+1}(y)$, and since, by hypothesis, $R_k(x) = R_k(y)$, we have that $R_{k+1}(x) \subseteq R_{k+1}(y)$, and, consequently, $R_{k+1}(x) \neq R_{k+1}(y)$, thus providing a contradiction to the hypothesis $R_{k+1}(x) = R_{k+1}(y)$, and therefore proving that $P_{k+1}(x) = P_{k+1}(y)$.

We now prove the dual result on Q. Assume by contradiction that $Q_{k+1}(x) \neq Q_{k+1}(y)$. This implies that the $k+1$-th iteration was a *Refine* one in which a triple $\langle a, B, C\rangle$ was selected to be stabilized and $Q_k(x) \in \tau_k(B)$. Moreover, since $Q_k(x)$ has been split, then both $Q_k(x) \cap \mathrm{pre}_a(\cup\tau_k(C)) \neq \varnothing$ and $Q_k(x) \not\subseteq \mathrm{pre}_a(\cup\tau_k(C))$ hold. Suppose $Q_{k+1}(x) = Q_k(x) \cap \mathrm{pre}_a(\cup\tau_k(C))$ (the case $Q_{k+1}(y) = Q_k(y) \cap \mathrm{pre}_a(\cup\tau_k(C))$ is symmetric). Thus, $Q_{k+1}(y) = Q_k(y) \setminus \mathrm{pre}_a(\cup\tau_k(C))$ holds. Consequently, line 19 will remove $Q_{k+1}(y)$ from $\tau_k(B')$, while $Q_{k+1}(x) \in \tau_{k+1}(B')$ will still hold. Thus, let $b \in B'$, so that $x \in \cup\tau_{k+1}(P_{k+1}(b))$ but $y \notin \cup\tau_{k+1}(P_{k+1}(b))$. This entails $b \in R^{-1}_{k+1}(x)$ and $b \notin R^{-1}_{k+1}(y)$, implying $R^{-1}_{k+1}(x) \neq R^{-1}_{k+1}(y)$, which gives a contradiction to the hypothesis $R^{-1}_{k+1}(x) = R^{-1}_{k+1}(y)$, thus proving that $Q_{k+1}(x) = Q_{k+1}(y)$. Finally, since Algorithm 4 terminates after t iterations then $P = P_t$ and $Q = Q_t$ and the following holds:

$$\big(\forall j \in [0,t].\, R_j(x) = R_j(y)\big) \Rightarrow P(x) = P(y)$$
$$\big(\forall j \in [0,t].\, R^{-1}_j(x) = R^{-1}_j(y)\big) \Rightarrow Q(x) = Q(y)$$

To conclude, (2PR.a) and (2PR.b) follow from the above results by taking the contrapositive. □

9 Extension to Infinite Transition Systems

In this section we extend the proofs for Theorems 5.3, 6.1 and 8.1 to infinite state systems, that is, we show how their proofs can be adapted to prove Theorems 5.6, 6.2 and 8.2. Specifically, we show that the three proofs can be extended under a convergence assumption on the following basic simulation algorithm Sim, which is a variation of the *SchematicSimilarity1* algorithm by Henzinger et al. [1995].

Algorithm 5: Sim

Input: A transition system $G = (\Sigma, I, L, \to)$, a relation $R_i \in \mathrm{Rel}(\Sigma)$ such that $R_{\mathrm{sim}} \subseteq R_i \subseteq R_{\mathrm{po}}$ for some preorder R_{po}, and the simulation preorder R_{sim} induced by R_{po}.

1 $\mathrm{Rel}(\Sigma) \ni R := R_i$;
2 **while** $\exists x, x' \in \Sigma, a \in L.\, x \xrightarrow{a} x' \wedge R(x) \not\subseteq \mathrm{pre}_a(R(x'))$ **do**
  // $\textsc{Inv}_1$: $\forall x \in \Sigma.\, R_{\mathrm{sim}}(x) \subseteq R(x) \subseteq R_i(x)$
3 $R(x) := R(x) \cap \mathrm{pre}_a(R(x'))$;
4 **return** R;

More formally, we state Assumption 9.1 as follows:

Assumption 9.1. If $y \in R_i(x) \setminus R_{\mathrm{sim}}(x)$ for some $x \in \Sigma$, then there exists an execution (not necessarily terminating) of the algorithm Sim over input R_i such that, after a finite number of iterations, $y \notin R(x)$ holds.

We observe that Assumption 9.1 is essentially an algorithmic reformulation of the first limit ordinal convergence requirement in the context of the well-known simulation approximants based on ordinals, that is, Assumption 9.1 is equivalent to Assumption 5.5. As a consequence, it is

known that Assumption 9.1 holds for all finitely branching systems [Hofman et al. 2016, paragraph following Definition 30].

We now proceed to extend the common structure underlying the proofs of Theorems 5.3, 6.1 and 8.1 to infinite state systems, when Assumption 5.5 (and thus Assumption 9.1) holds, and since the three proofs follow a common structure, we will comment on them in parallel.

We first note that the only points in the proofs where finiteness of the state space is relied upon are, respectively, point (iv) of Theorem 5.3, point (iii) of Theorem 6.1 and point (iv) of Theorem 8.1. Therefore, we show how their common pattern can be generalized to transition systems with infinitely many states (in the following we refer to the notation used in the respective proofs). Observe that, if the input transition system is finite, then the execution of Sim over the output relation R is guaranteed to terminate, and correctness of Sim ensures that an iteration *it* where some element of $\mathbb{S}$ is refined will be eventually reached in finite time, for every execution (regardless of how nondeterminism is resolved). On the other hand, for transition systems with infinitely many states, an iteration such as *it* might not exist, since Sim might not terminate on infinite state systems and, moreover, nondeterminism might choose to refine principals only if they are not marked as reachable (that is, $R(x) \cap \sigma = \varnothing$ for Theorem 5.3, $\nexists s \in \sigma.\, R(x) = R(s)$ for Theorem 6.1, and $\cup\tau(B) \cap \sigma = \varnothing$ for Theorem 8.1), and which are, therefore, not included in $\mathbb{S}$. However, we can use Assumption 9.1 to avoid this situation altogether since, assuming $\mathbb{S} \neq \varnothing$ (as we do, by contradiction, in the proofs), we can pick a principal $R(z) \in \mathbb{S}$ and some $z' \in R(z) \setminus R_{\text{sim}}(z) \neq \varnothing$ by definition of $\mathbb{S}$. Assumption 9.1 therefore ensures that there exists an execution where Sim reaches an iteration it_1 where z' is removed from the principal of z. Then, let R_1 be the current relation at iteration $it_1 + 1$ and define $\mathbb{S}' = \{R(x) \mid x \in \Sigma,\, R(x) \in \mathbb{S},\, R_1(x) \neq R(x)\}$ to be the set of principals which have been refined by Sim up to iteration $it_1 + 1$. We find that $\mathbb{S}'$ is nonempty by construction since $R(z) \in \mathbb{S}'$, and that $R_{\text{sim}} \subseteq R_1 \subseteq R$ following Inv_1. The proof then proceeds as explained in the respective points (iv) of Theorem 5.3, (iii) of Theorem 6.1 and (iv) of Theorem 8.1, by considering the first principal in $\mathbb{S}'$ to be refined and the corresponding iteration *it*, allowing us to get a contradiction, thus proving that $\mathbb{S} = \varnothing$.

10 Conclusion and Future Work

We introduced and proved the correctness and termination of algorithms for the reachable simulation problem. We showed fundamental differences w.r.t. to the analogous problem for bisimulation studied by Lee and Yannakakis [1992]. To the best of our knowledge, this is the first investigation of Lee and Yannakakis' problem recast to the simulation preorder and partition. Algorithm 4 is the most relevant one for practical purposes, since this procedure converges on all finite state systems and is well-suited to handle infinite state systems through its symbolic representation, being able to converge on some—but not all, due to undecidability—such infinite systems. In particular, we have shown that Algorithm 4 offers the same termination guarantee as the LY algorithm [Lee and Yannakakis 1992]. On top of that, our algorithm also terminates under a local finiteness hypothesis, while LY has no counterpart to such a termination guarantee.

Future work will explore possible domains in which the algorithm can be applied. Choosing a specific class of implicitly represented systems gives rise to a multitude of questions which are domain dependent and solving these are crucial to obtain an efficient implementation—note that reachability analysis is mostly superfluous on explicitly represented systems as they usually do not encode unreachable states. Orthogonally, further generalizations of our algorithm, for instance, to other behavioral relations, constitute future research paths. One appealing relation is branching bisimilarity for labeled transition systems or stuttering equivalence for Kripke structures, which have been proven in [Groote et al. 2017; Jansen et al. 2020] to be symbolically computable in $O(|{\to}| \log |\Sigma|)$

time, the same complexity of the renowned Paige-Tarjan bisimulation algorithm [Paige and Tarjan 1987].

Acknowledgments

Nicolas Manini was partially supported by the grant PIPF-2022/COM-24370, funded by the Madrid Regional Government. Francesco Ranzato was partially funded by a *WhatsApp Research Award* on "Privacy-aware Program Analysis", an *Amazon Research Award* for *AWS Automated Reasoning* on "Implicit Program Analysis", and by the *Italian MUR*, under the PRIN 2022 PNRR project no. P2022HXNSC. This publication is part of the grant PID2022-138072OB-I00, funded by MCIN, FEDER and UE. Finally, this work was partially supported by the PRODIGY Project (TED2021-132464B-I00) funded by MCIN and the European Union NextGeneration and by the ESF.

References

Rajeev Alur and Thomas A. Henzinger. 1999. Computer-Aided Verification. (1999). Chapter 4: Graph Minimization. (Unpublished manuscript).

Saddek Bensalem, Ahmed Bouajjani, Claire Loiseaux, and Joseph Sifakis. 1992. Property Preserving Simulations. In *Proc. of the Fourth International Workshop on Computer Aided Verification, CAV'92 (LNCS)*. Springer, 260–273. https://doi.org/10.1007/3-540-56496-9_21

Bard Bloom and Robert Paige. 1995. Transformational Design and Implementation of a New Efficient Solution to the Ready Simulation Problem. *Sci. Comput. Program.* 24, 3 (1995), 189–220. https://doi.org/10.1016/0167-6423(95)00003-B

Ahmed Bouajjani, Jean-Claude Fernandez, and Nicolas Halbwachs. 1991. Minimal Model Generation. In *Proc. of the International Workshop on Computer-Aided Verification, CAV'91 (LNCS)*. Springer, 197–203. https://doi.org/10.1007/BFb0023733

Ahmed Bouajjani, Jean-Claude Fernandez, Nicolas Halbwachs, Pascal Raymond, and Christophe Ratel. 1992. Minimal State Graph Generation. *Science of Computer Programming* 18, 3 (1992), 247–269. https://doi.org/10.1016/0167-6423(92)90018-7

Doron Bustan and Orna Grumberg. 2003. Simulation-based minimization. *ACM Trans. Comput. Log.* 4, 2 (2003), 181–206. https://doi.org/10.1145/635499.635502

Gérard Cécé. 2017. Foundation for a series of efficient simulation algorithms. In *Proc. of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*. IEEE Computer Society, 1–12. https://doi.org/10.1109/LICS.2017.8005069

Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. 2018. *Handbook of Model Checking* (1st ed.). Springer.

Silvia Crafa, Francesco Ranzato, and Francesco Tapparo. 2011. Saving Space in a Time Efficient Simulation Algorithm. *Fundam. Informaticae* 108, 1-2 (2011), 23–42. https://doi.org/10.3233/FI-2011-412

Kathi Fisler and Moshe Y. Vardi. 2002. Bisimulation Minimization and Symbolic Model Checking. *Formal Methods Syst. Des.* 21, 1 (2002), 39–78. https://doi.org/10.1023/A:1016091902809

Pierre Ganty, Nicolas Manini, and Francesco Ranzato. 2024. Computing Reachable Simulations on Transition Systems. In *Proceedings of the 18th International Conference on Reachability Problems, RP 2024 (Lecture Notes in Computer Science, Vol. 15050)*. Springer, 21–37. https://doi.org/10.1007/978-3-031-72621-7_3

Raffaella Gentilini, Carla Piazza, and Alberto Policriti. 2003. From bisimulation to simulation: Coarsest partition problems. *Journal of Automated Reasoning* 31, 1 (2003), 73–103. https://doi.org/10.1023/A:1027328830731

Rob van Glabbeek and Bas Ploeger. 2008. Correcting a space-efficient simulation algorithm. In *Proc. of the International Conference on Computer Aided Verification, CAV'08*. Springer, 517–529. https://doi.org/10.1007/978-3-540-70545-1_49

George A Gratzer. 2011. *Lattice theory: foundation*. Springer.

Jan Friso Groote, David N. Jansen, Jeroen J. A. Keiren, and Anton J. Wijs. 2017. An $O(m \log n)$ Algorithm for Computing Stuttering Equivalence and Branching Bisimulation. *ACM Trans. Comput. Logic* 18, 2, Article 13 (jun 2017), 34 pages. https://doi.org/10.1145/3060140

Orna Grumberg and David E. Long. 1991. Model Checking and Modular Verification. In *Proc. of the 2nd International Conference on Concurrency Theory, CONCUR'91 (LNCS)*. Springer, 250–265. https://doi.org/10.1007/3-540-54430-5_93

Orna Grumberg and David E Long. 1994. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 16, 3 (1994), 843–871. https://doi.org/10.1145/177492.177725

Bhargav S. Gulavani, Thomas A. Henzinger, Yamini Kannan, Aditya V. Nori, and Sriram K. Rajamani. 2006. SYNERGY: a new algorithm for property checking. In *Proc. of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006*. ACM, 117–127. https://doi.org/10.1145/1181775.1181790

Monika Rauch Henzinger, Thomas A. Henzinger, and Peter W. Kopke. 1995. Computing Simulations on Finite and Infinite Graphs. In *Proc. of IEEE 36th Annual Foundations of Computer Science, FOCS'95*. 453–462. https://doi.org/10.1109/SFCS.

1995.492576

Thomas A. Henzinger and Peter W Kopke. 1995. *Hybrid Automata with Finite Mutual Simulations.* Technical Report TR-95-1497. Computer Science Departement.

Thomas A. Henzinger, Rupak Majumdar, and Jean-François Raskin. 2005. A Classification of Symbolic Transition Systems. *ACM Transactions on Computational Logic* 6, 1 (2005), 1–32. https://doi.org/10.1145/1042038.1042039

Piotr Hofman, Slawomir Lasota, Richard Mayr, and Patrick Totzke. 2016. Simulation problems over one-counter nets. *Logical Methods in Computer Science* 12 (2016). https://doi.org/10.2168/LMCS-12(1:6)2016

David N. Jansen, Jan Friso Groote, Jeroen J. A. Keiren, and Anton Wijs. 2020. An $O(m \log n)$ algorithm for branching bisimilarity on labelled transition systems. In *Proceedings of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2020 (Lecture Notes in Computer Science, Vol. 12079)*. Springer, 3–20. https://doi.org/10.1007/978-3-030-45237-7_1

Antonín Kučera and Petr Jančar. 2006. Equivalence-checking on infinite-state systems: Techniques and results. *Theory and Practice of Logic Programming* 6, 3 (2006), 227–264. https://doi.org/10.1017/S1471068406002651

Antonín Kucera and Richard Mayr. 2002a. Simulation Preorder over Simple Process Algebras. *Inf. Comput.* 173, 2 (2002), 184–198. https://doi.org/10.1006/inco.2001.3122

Antonín Kucera and Richard Mayr. 2002b. Why Is Simulation Harder than Bisimulation?. In *Proc. of the 13th International Conference on Concurrency Theory, CONCUR 2002 (LNCS)*. Springer, 594–610. https://doi.org/10.1007/3-540-45694-5_39

David Lee and Mihalis Yannakakis. 1992. Online Minimization of Transition Systems. In *Proc. of the 24th Annual ACM Symposium on Theory of Computing, STOC '92*. ACM, 264–274. https://doi.org/10.1145/129712.129738

Claire Loiseaux, Susanne Graf, Joseph Sifakis, Ahmed Bouajjani, and Saddek Bensalem. 1995. Property Preserving Abstractions for the Verification of Concurrent Systems. *Formal Methods Syst. Des.* 6, 1 (1995), 11–44. https://doi.org/10.1007/BF01384313

Rupak Majumdar, Necmiye Ozay, and Anne-Kathrin Schmuck. 2020. On Abstraction-Based Controller Design with Output Feedback. In *Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control, HSCC 2020*. ACM, 1–11. https://doi.org/10.1145/3365365.3382219

Marvin L. Minsky. 1967. *Computation: Finite and Infinite Machines.* Prentice-Hall.

Robert Paige and Robert E. Tarjan. 1987. Three Partition Refinement Algorithms. *SIAM J. Comput.* 16, 6 (1987), 973–989. https://doi.org/10.1137/0216062

Corina S. Pasareanu, Radek Pelánek, and Willem Visser. 2005. Concrete Model Checking with Abstract Matching and Refinement. In *Proc. 17th International Conference on Computer Aided Verification, CAV 2005 (LNCS)*. Springer, 52–66. https://doi.org/10.1007/11513988_7

Francesco Ranzato. 2013. A More Efficient Simulation Algorithm on Kripke Structures. In *Proc. of the 38th International Symposium on Mathematical Foundations of Computer Science 2013, MFCS 2013 (LNCS)*. Springer, 753–764. https://doi.org/10.1007/978-3-642-40313-2_66

Francesco Ranzato. 2014. An efficient simulation algorithm on Kripke structures. *Acta informatica* 51, 2 (2014), 107–125. https://doi.org/10.1007/s00236-014-0195-9

Francesco Ranzato and Francesco Tapparo. 2007. A New Efficient Simulation Equivalence Algorithm. In *Proc. of the 22nd IEEE Symposium on Logic in Computer Science, LICS 2007*. IEEE Computer Society, 171–180. https://doi.org/10.1109/LICS.2007.8

Francesco Ranzato and Francesco Tapparo. 2010. An Efficient Simulation Algorithm Based on Abstract Interpretation. *Information and Computation* 208, 1 (2010), 1–22. https://doi.org/10.1016/j.ic.2009.06.002

Li Tan and Rance Cleaveland. 2001. Simulation Revisited. In *Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001 (LNCS)*. Springer, 480–495. https://doi.org/10.1007/3-540-45319-9_33

Rob van Glabbeek and Bas Ploeger. 2008. Five Determinisation Algorithms. In *Implementation and Applications of Automata (LNCS)*. Springer, 161–170. https://doi.org/10.1007/978-3-540-70844-5_17

Mihalis Yannakakis and David Lee. 1997. An Efficient Algorithm for Minimizing Real-Time Transition Systems: Extended Abstract. *Formal Methods in System Design* 11, 2 (1997), 113–136. https://doi.org/10.1023/A:1008621829508

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009