Computing Reachable Simulations on Transition Systems

Pierre Ganty[1] [0000−0002−3625−6003], Nicolas Manini✉[1,2] [0000−0002−7561−3763], and Francesco Ranzato[3] [0000−0003−0159−0068]

[1] IMDEA Software Institute, Pozuelo de Alarcón, Spain
{pierre.ganty,nicolas.manini}@imdea.org
[2] Universidad Politécnica de Madrid, Madrid, Spain
[3] University of Padova, Padova, Italy
francesco.ranzato@unipd.it

Abstract. We study the problem of computing the reachable principals of the simulation preorder and the reachable blocks of simulation equivalence. Following a theoretical investigation of this problem, which highlights a sharp contrast with the already settled case of bisimulation, we design algorithms to solve this 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. In particular, we put forward a symbolic algorithm processing state partitions and, in turn, relations between their blocks, which is suited for processing infinite-state systems.

Keywords: Simulation · Reachability · Reachable Simulation Problem · Simulation algorithm · Symbolic Data Structure.

1 Introduction

Given a possibly infinite labeled transition system S, we study the problem of computing the reachable principals[1] of the greatest simulation preorder R_{sim}, and the reachable blocks of the induced simulation partition P_{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 system's reachable states, 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 requires the computation of the entire simulation preorder and, possibly, of all the reachable states. Here, we present a completely different solution relying on a convoluted interleaving of reachability and simulation computation, possibly avoiding the computation of all the reachable states and of the whole R_{sim}.

[1] The term principal comes from the well-known notion of *principal ideal* [13, Chapter I, sect. 3.4]. Detailed definitions are given in Section 3.

Contributions. We study the reachable simulation problem by showing, through an unsolvability proof (cf. Section 4), that there is a stark contrast w.r.t. the problem of computing the reachable blocks of the bisimulation partition, settled by Lee and Yannakakis [23] in STOC 1992. In Section 5, we put forward an algorithm computing the reachable part of the simulation preorder, and yielding an over-approximation for the partition. We prove correctness and termination on finite state systems, and extend correctness (under simple assumptions) to infinite state systems. Moreover, we provide examples showing termination on some infinite state systems[2]. Section 6 introduces the so-called 2PR (2 state Partitions and a Relation among them) triples which we use to design a symbolic algorithm for the reachable simulation problem. 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 that is quadratic in the number of blocks of (a portion of) the bisimulation partition. As an auxiliary contribution, we define partitions induced by arbitrary relations (not limited to preorders or equivalences), generalizing previous definitions.

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 [32]. 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 [32]. In a different context, the reachable blocks of the simulation partition define the states of the reduced quotiented system. The question of computing the transitions between the blocks of the reduced system has been investigated in depth by Bustan and Grumberg [6], who explore the difference and trade-off of the $\exists\exists$ (i.e., $B \rightarrow^{\exists\exists} B'$ iff $\exists s \in B.\, \exists s' \in B'.\, s \rightarrow s'$) and $\forall\exists$ definitions (i.e., $B \rightarrow^{\forall\exists} B'$ iff $\forall s \in B.\, \exists s' \in B'.\, s \rightarrow s'$).

Furthermore, solutions to the reachable *simulation* problem have potential applications in program and hybrid systems verification, as past research [16, 25, 26, 33] has leveraged solutions to the reachable *bisimulation* problem.

Related Work. The closest work to ours is that by Lee and Yannakakis [23], who first designed an interleaving of reachability and bisimulation computation, here referred to as the LY algorithm. Their work is highly cited and has been applied and revisited several times (e.g. [1, 10]), nevertheless, it remains

[2] As shown in Section 4, an algorithm terminating on all infinite state systems cannot exist.

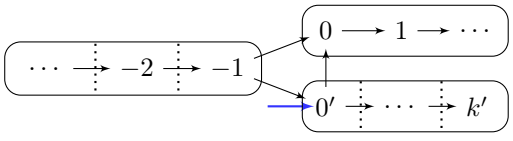
an elaborate algorithm with hidden subtleties. Let us point out that we were not able to find, for some results claimed in the original paper [23], any proof argument (either searching online or contacting authors). To put the LY algorithm in its historical context, it was one of several algorithms to compute the reachable part of the bisimulation-based quotiented system [4, 5, 23]. These algorithms share the interleaving of the bisimulation computation—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 (possibly more often) as the naïve procedure consisting in first computing the bisimulation and next determining its reachable blocks. Later on, Alur and Henzinger revisited these algorithms for reachable bisimulation in their unpublished book on computer-aided verification [1, Chapter 4], while Fisler and Vardi conducted a theoretical and experimental evaluation of the LY algorithm [10]. 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 [16, 26] to hybrid systems verification, where [25] employs 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 [2, 6, 8, 14, 15, 24]. Moreover, infinite state systems like 2D rectangular automata may have infinite bisimilarity quotients, yet they always have finite similarity quotients (see [17, 18]). There is a large body of work [3, 7, 9, 11, 12, 17, 27, 28, 29, 30, 31] on efficiently computing the simulation preorder, through both explicit or symbolic algorithms. Kucera and Mayr [21, 22] 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. Due to lack of space, some material (e.g., proofs) is omitted.

2 Motivating Example

We introduce an example showing the challenges of the reachable simulation problem and some intuitive explanations on our solution. Consider the family of infinite transition systems parameterized over an integer $k \geq 0$ depicted below.



The set of states is $\mathbb{Z} \cup \{0', \ldots, k'\}$, and the transition relation is given by the arrows in the diagram. The initial state is $0'$, denoted by the incoming blue arrow.

Consider the initial partition of states given by the three 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. Let us remark three observations about this family of transition systems: (1) there are infinitely many reachable states; (2) there are infinitely many simulation equivalence classes, depicted by dotted lines splitting the boxes; 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 paper, we tackle the challenge of effectively computing information such as in (3). 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 would not terminate) or refining the blocks of the initial partition to the simulation partition (a simulation algorithm would not terminate). Yet, in this work, we define algorithms alternating bounded state space exploration and partition refinement. Such algorithms 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 5.1 shows a run of our algorithm on this example.

3 Background

Preorders 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; $\mathrm{Rel}(\Sigma) \ni R^{-1} \triangleq \{(y, x) \in \Sigma \times \Sigma \mid (x, y) \in R\}$ is the *converse* relation of R. Moreover, for a given set $S \subseteq \Sigma$, we denote by $R^S \triangleq \{R(x) \in \wp(\Sigma) \mid x \in \Sigma, R(x) \cap S \neq \varnothing\}$ 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. More in general, 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 given by $R \cap R^{-1}$. These induced partitions will be a key ingredient of our approach.

Simulation and Bisimulation. Let $G = (\Sigma, I, L, \to)$ 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 $\to \subseteq \Sigma \times L \times \Sigma$ is the *labeled transition relation*, where we denote $(x, a, y) \in \to$ as $x \xrightarrow{a} y$. When L is a singleton set or when the label is unimportant we simply write $x \to y$. This comes handy in our examples where we assume L is a singleton. Given $a \in L$, $\mathsf{post}_a \colon \wp(\Sigma) \to \wp(\Sigma)$ denotes the usual successor transformer $\mathsf{post}_a(X) \triangleq \{y \in \Sigma \mid \exists x \in X.\, x \xrightarrow{a} y\}$, and, dually, $\mathsf{pre}_a \colon \wp(\Sigma) \to \wp(\Sigma)$ is the predecessor $\mathsf{pre}_a(Y) \triangleq \{x \in \Sigma \mid \exists y \in Y.\, x \xrightarrow{a} y\}$. Moreover, we define $\mathsf{post} \colon \wp(\Sigma) \to \wp(\Sigma)$ as $\mathsf{post}(X) \triangleq \cup_{a \in L} \mathsf{post}_a(X)$ and, symmetrically, $\mathsf{pre} \colon \wp(\Sigma) \to \wp(\Sigma)$ as $\mathsf{pre}(X) \triangleq \cup_{a \in L} \mathsf{pre}_a(X)$. Thus, $\mathsf{post}^*(I) \triangleq \cup_{n \in \mathbb{N}} \mathsf{post}^n(I)$ 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 $s \xrightarrow{a} s'$, $R(s)$ is *a-stable* (or simply *stable*) w.r.t. $R(s')$ when $R(s) \subseteq \mathsf{pre}_a(R(s'))$, 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 $R(s) \cap \mathsf{pre}_a(R(s'))$. 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*[3] (or similarity). 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*.

Remark 3.1 (On the Initial Preorder). We point out that the role of the 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 [32]). For the bisimulation case, such a priori information is conveyed by an equivalence—e.g. specified by 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.

[3] 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 $R_{\mathrm{sim}} \cap (R_{\mathrm{sim}})^{-1}$.

4 The Reachable Simulation Problem

We formally define the problem investigated in this work, which extends in a natural way the reachable bisimulation problem tackled by Lee and Yannakakis in [23].

*Problem 4.1 (**The Reachable Simulation Problem**).*
GIVEN: A labeled transition system $G = (\Sigma, I, L, \rightarrow)$ and $R_i \in \mathrm{PreO}(\Sigma)$.
COMPUTE: The reachable principals of R_{sim} and the reachable blocks of P_{sim}.

A first challenge 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 the set of blocks containing at least one reachable state, that is $P^{\mathsf{post}^*(I)}$. Thus, the following equality holds:

$$P^{\mathsf{post}^*(I)} = \{B \in P \mid B \cap \mathsf{post}^*(I) \neq \varnothing\} = \{P(s) \in P \mid s \in \mathsf{post}^*(I)\} \ . \quad \text{(rb)}$$

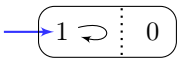
Moving from bisimulation to simulation, (rb) can be generalized to any of the two following definitions of *reachable principal* (rp) of a reflexive relation $R \in \mathrm{Rel}(\Sigma)$, which can both be deemed adequate:

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

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

Clearly, when R is an equivalence relation, (rp_1) and (rp_2) coincide and boil down to (rb). In general, only $R_{\mathrm{alt}}^{\mathsf{post}^*(I)} \subseteq R^{\mathsf{post}^*(I)}$ holds, and, moreover, the inclusion $(R_{\mathrm{sim}})_{\mathrm{alt}}^{\mathsf{post}^*(I)} \subsetneq (R_{\mathrm{sim}})^{\mathsf{post}^*(I)}$ can hold strictly[4], as shown next.

Example 4.2. Consider the system depicted below, its initial preorder $R_i = \{0,1\} \times \{0,1\}$, and the block induced by R_i, represented as a box.



The simulation preorder w.r.t. R_i is $R_{\mathrm{sim}}(0) = \{0,1\}$, $R_{\mathrm{sim}}(1) = \{1\}$, and a dotted line delimits the blocks of P_{sim}. Thus, since $\mathsf{post}^*(I) = \{1\}$, we get that the reachable principals as per (rp_2) are the singleton $\{R_{\mathrm{sim}}(1)\}$, while the reachable principals as per (rp_1) are $\{R_{\mathrm{sim}}(0), R_{\mathrm{sim}}(1)\}$. ◇

Unsolvability. 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 the reachable blocks of P_{sim} is unsolvable. This result shows a difference with the problem of computing reachable blocks of P_{bis}, which Lee and Yannakakis [23] proved solvable for finite bisimulations.

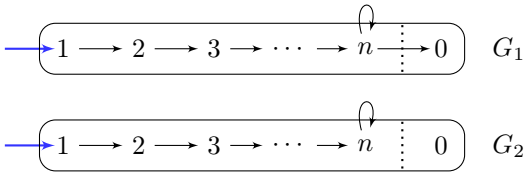
[4] For some systems, (rp_1) could even be infinite, and (rp_2) be a finite set.

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 holds since solvability of Problem 4.1 would imply 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 [20].

Remarks for Finite State Systems. We observe a further difference between Problem 4.1 and the corresponding problem for bisimulation, 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 $\mathsf{post}^*(I)$ and P_{bis} or P_{sim}, and, check whether $\mathsf{post}^*(I) \cap B = \varnothing$ holds for every block B in P_{bis} or P_{sim}. However, for the case of bisimulation, deciding reachability of a block $B \in P_{\mathrm{bis}}$ can be done in $O(|P_{\mathrm{bis}}^{\mathsf{post}^*(I)}|)$ time by leveraging the definition of bisimulation: if x and y are bisimilar states, then x can reach B in one transition iff y can, hence picking any single state per block of P_{bis} suffices. For the simulation case, deciding the reachability of $B \in P_{\mathrm{sim}}$ is more involved: as the following example hints, to decide whether a block $B \in P_{\mathrm{sim}}$ is reachable we possibly have to check whether there is a path of arbitrary length (that is, in $O(|\Sigma|)$) in the transition system reaching B, so that no bound on the number of blocks of P_{sim} applies.

Example 4.4. Let G_1, G_2 be the transition systems depicted below.



Blocks of states sharing the same principal in $R_i = \mathbb{N} \times \mathbb{N}$ are represented as boxes, while dotted lines are used to delimit the blocks of P_{sim} w.r.t. R_i. In fact, we have that $R_{\mathrm{sim}}(0) = [0, n]$, and, for all $k \in [1, n]$, $R_{\mathrm{sim}}(k) = [1, n]$, so we have that $P_{\mathrm{sim}} = \{[0, 0], [1, n]\}$ for G_1 and G_2. Observe that in G_1 the block $[0, 0] \in P_{\mathrm{sim}}$ is reachable while in G_2 the block $[0, 0] \in P_{\mathrm{sim}}$ is unreachable. Hence, to decide whether $[0, 0]$ is reachable in these two systems, we have to detect that 0 is reachable in G_1 and not in G_2. However, this is not the case for bisimulation, since it turns out that $P_{\mathrm{bis}} = \{[k, k]\}_{k=0,\ldots,n}$ for G_1, while $P_{\mathrm{bis}} = P_{\mathrm{sim}}$ for G_2. $\diamond$

5 A Reachable Simulation Algorithm

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 reachable blocks of P_{sim}.

Algorithm 1: Relation-based Algorithm

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

1 $\mathrm{Rel}(\Sigma) \ni R \coloneqq R_i$; $\quad \wp(\Sigma) \ni \sigma \coloneqq \sigma_i$;
2 **while true do**
  // $\textsc{Inv}_1$: $\forall x \in \Sigma.\, R_{\mathrm{sim}}(x) \subseteq R(x) \subseteq R_i(x)$
  // $\textsc{Inv}_2$: $\sigma_i \subseteq \sigma \subseteq \mathsf{post}^*(I)$ $\quad$ $\textsc{Inv}_3$: $\forall x \in \Sigma.\, x \in R(x)$
3 $U \coloneqq \{R(x) \mid R(x) \cap \sigma = \varnothing,\ R(x) \cap (I \cup \mathsf{post}(\sigma)) \neq \varnothing\}$;
4 $V \coloneqq \{\langle a, x, x' \rangle \in L \times \Sigma^2 \mid R(x) \cap \sigma \neq \varnothing, x \xrightarrow{a} x',\ R(x) \nsubseteq \mathsf{pre}_a(R(x'))\}$;
5 **nif**
6 $(U \neq \varnothing) \longrightarrow$ *Search* :
7 **choose** $R(x) \in U$, $s \in R(x) \cap (I \cup \mathsf{post}(\sigma))$;
8 $\sigma \coloneqq \sigma \cup \{s\}$;
9 $(V \neq \varnothing) \longrightarrow$ *Refine* :
10 **choose** $\langle a, x, x' \rangle \in V$;
11 $R(x) \coloneqq R(x) \cap \mathsf{pre}_a(R(x'))$;
12 $(U = \varnothing \wedge V = \varnothing) \longrightarrow$ **return** $\langle R, \sigma \rangle$;

This algorithm maintains a relation $R \in \mathrm{Rel}(\Sigma)$ specified through its principals $R(x) \in \wp(\Sigma)$, and a set $\sigma \subseteq \Sigma$ of reachable states[5], so that $R^\sigma = \{R(x) \mid R(x) \cap \sigma \neq \varnothing\}$ are the provably reachable principals of R. The algorithm computes the set U of principals which can be added to R^σ and the set V of unstable principal pairs. 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 $R(x) \nsubseteq \mathsf{pre}_a(R(x'))$. 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. Details on the specific representations are given in Section 6.

Algorithm 1 either updates, in the *Search* block, the reachability information, 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 6–8) or the *Refine* blocks (lines 9–11) are executed, or, at line 12, when the other guards are 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.

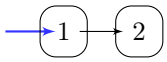
A principal $R(x)$ is refined at line 11 provided it is provably reachable. Upon termination, the principals of R^σ and the principals of $R_{\mathrm{sim}}^{\mathsf{post}^*(I)}$ coincide (cf. (1.a) of Theorem 5.3 below)[6]. However, R may well contain unstable principals, so

[5] We distinguish the states of $\mathsf{post}^*(I)$ from those in its subset σ by referring to the states in σ as *provably reachable*. We extend this notion to principals.

[6] Note that line 11 might break transitivity of R. In fact, R is not guaranteed to be a preorder during execution, and not even at termination.

that, in general, R and R_{sim} do not coincide. 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 $P_{\mathrm{sim}}^{\mathsf{post}^*(I)}$. In fact, P^{σ} might lack some of the blocks of $P_{\mathrm{sim}}^{\mathsf{post}^*(I)}$.

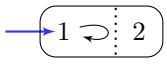
Example 5.1. Consider the transition system depicted below, where $R_i = \{(1,1),(2,1),(2,2)\} = R_{\mathrm{sim}}$, and $\sigma_i = \varnothing$.



We have that $\mathsf{post}^*(I) = \{1,2\}$, and the blocks induced by R_i coincide with $P_{\mathrm{sim}} = \{\{1\},\{2\}\}$ and are depicted as boxes. Algorithm 1 on input R_i and σ_i returns $\langle R_{\mathrm{sim}}, \{1\}\rangle$. Therefore, it turns out that $P_{\mathrm{sim}}^{\mathsf{post}^*(I)} = P_{\mathrm{sim}}$ and $P^{\sigma} = \{\{1\}\}$, so that $P_{\mathrm{sim}}^{\mathsf{post}^*(I)} \not\subseteq P^{\sigma}$ holds. ◇

Algorithm 1 aims at populating σ with just enough states to correctly characterize the reachable principals (i.e., achieving (1.a)), but such states are, in general, not enough to intersect all the reachable blocks of P_{sim}. 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 $\{B \in P \mid R(B) \cap \sigma \neq \varnothing\}$ as defined in (1.b). Observe that reachable blocks are computed precisely: each block of $P_{\mathrm{sim}}^{\mathsf{post}^*(I)}$ is in P, but, in general, not all blocks in P belong to P_{sim}. Moreover, the following example shows that the converse inclusion of (1.b) does not always hold.

Example 5.2. Consider the system depicted below with $\mathsf{post}^*(I) = \{1\}$, $R_i = \{1,2\} \times \{1,2\}$, $R_{\mathrm{sim}} = \{(1,1),(2,1),(2,2)\}$, and $P_{\mathrm{sim}} = \{\{1\},\{2\}\}$, where boxes depict blocks induced by R_i and dotted lines delimit the blocks of P_{sim}.



Algorithm 1 on input R_i and $\sigma_i = \varnothing$ outputs $R = R_{\mathrm{sim}}$ and $\sigma = \{1\}$. Thus, the inclusion of (1.b) is strict since $\{\{1\}\} \subsetneq \{\{1\},\{2\}\}$. ◇

On the other hand, this inclusion is not arbitrarily loose since a block $B \in P$ such that $R(B) \cap \sigma \neq \varnothing$ (cf. (1.b)) is guaranteed to be simulated by some reachable state. Therefore, R_{sim} limits the magnitude of this over-approximation.

It turns out that Algorithm 1 is correct and terminates on finite state systems.

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

$$R_{\mathrm{sim}}^{\mathsf{post}^*(I)} = R^{\sigma}, \quad \text{(1.a)} \qquad P_{\mathrm{sim}}^{\mathsf{post}^*(I)} \subseteq \{B \in P \mid R(B) \cap \sigma \neq \varnothing\}\ . \quad \text{(1.b)}$$

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 $\sigma_i \subseteq \mathsf{post}^*(I)$. Then, Algorithm 1 terminates on input G, R_i, and σ_i.*

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.

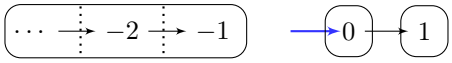
Assumption 5.5 (ω-Convergence for Simulation Approximants). *Given a transition system $G = (\Sigma, I, L, \to)$, and an initial preorder $R_i \in \mathrm{PreO}(\Sigma)$, let $\preceq_0 \triangleq R_i$ and $\preceq_n$, for $n > 0$, be the strong simulation approximants as defined in [19, Points (2) and (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 [19, Paragraph following Definition 30]. We can now formally state the correctness result for Algorithm 1 (on 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 $\sigma_i \subseteq \mathsf{post}^*(I)$. 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.*

Concerning termination, we point out that Algorithm 1 can terminate on some infinite systems, such as the one of Section 2, and the following example.

Example 5.7. Consider the infinite transition system depicted below, where $R_i(0) = \{0\}$, $R_i(1) = \{0, 1\}$, and for all $n < 0$, $R_i(n) =]-\infty, -1]$. As usual, boxes denote the blocks induced by R_i.

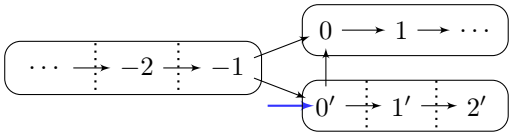


The simulation preorder is therefore: $R_{\mathrm{sim}}(0) = \{0\}$, $R_{\mathrm{sim}}(1) = \{0, 1\}$, and $R_{\mathrm{sim}}(n) =]-\infty, n]$ for each $n < 0$. Notice that R_{sim} has infinitely many principals, and the corresponding blocks of P_{sim} are delimited by dotted lines in the above diagram. After two *Search* iterations of Algorithm 1 (on input R_i, $\sigma_i = \varnothing$), we get $\sigma = \{0, 1\}$. At this point, $V = \varnothing$ and $U = \varnothing$ holds, and the algorithm returns the correct result.

Consider instead the process of refining each principal $R_i(x)$ such that $R_i(x) \neq R_{\mathrm{sim}}(x)$. This process, which converges to R_{sim}, cannot terminate after finitely many steps since infinitely many principals would need to be refined. ⋄

5.1 Execution on the Motivating Example

Let us run Algorithm 1 on the motivating example of Section 2. By fixing $k = 2$ we obtain the infinite state transition system depicted below.



The initial relation is given by $R_i(x) = \mathbb{Z} \setminus \mathbb{N}$ for all $x < 0$, $R_i(x) = \mathbb{N}$ for all $x \geq 0$, and $R_i(0') = R_i(1') = R_i(2') = \{0', 1', 2'\}$. As usual, states sharing the same principal in R_i are depicted in boxes, while dotted lines delimit the blocks of P_{sim}. The remaining input is set to $\sigma_i = \varnothing$.

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 $\sigma = \varnothing$, we have that $V = \varnothing$ holds. Hence, the *Search* block is executed and σ is updated: at line 8 we get $\sigma = \{0'\}$.
2nd iteration: Since $\sigma = \{0'\}$, we get $U = \{R_i(0)\} = \{\mathbb{N}\}$ because $\mathbb{N} \cap \mathsf{post}(\sigma) \neq \varnothing$. Moreover, $V = \{\langle a, 0', 1'\rangle, \langle a, 1', 2'\rangle\}$ since $\mathsf{pre}_a(R_i(1')) = \mathsf{pre}_a(R_i(2')) = \{0', 1'\}$, 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 using the principal in U: at line 8 we get $\sigma = \{0', 0\}$.
3rd iteration: We get $U = \varnothing$ since all the states reachable from σ are in principals intersecting σ. Moreover, V is as in the previous iteration. Note that $R_i(0)$, which now intersects σ, 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), and executes the *Refine* block: at line 11 the relation is updated, so that $R(1') = R_i(1') \cap \mathsf{pre}_a(R_i(2')) = \{0', 1'\}$.
4th iteration: Still, $U = \varnothing$, and $V = \{\langle a, 0', 1'\rangle\}$, since $R(0') = R_i(0') = \{0', 1', 2'\} \not\subseteq \mathsf{pre}_a(R(1')) = \{0'\}$. Therefore, the algorithm executes a *Refine* step, and at line 11 we get $R(0') = \{0', 1', 2'\} \cap \{0'\} = \{0'\}$.
5th iteration: Again $U = \emptyset$ since $\sigma = \{0', 0\}$, and for every $x \in \mathbb{N} \cup \{0', 1', 2'\}$, it holds $R(x) \cap \sigma \neq \varnothing$. 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 as follows: $R(0') = \{0'\}$, $R(1') = \{0', 1'\}$, and $R(x) = R_i(x)$ for every other state. Observe that $|\sigma| = 2$, independently of the fixed parameter k. In fact, Algorithm 1 explores two reachable states (out of infinitely many) which suffice to characterize all the reachable principals.

6 2PR Triples for Designing a Symbolic Algorithm

Symbolic approaches for simulation algorithms based on state partitions are beneficial for algorithms manipulating infinite state systems, as shown by Henzinger et al.'s [17] symbolic simulation algorithm for infinite graphs, and, in particular, hybrid automata. Symbolic approaches are also advantageous in terms of space and time efficiency for finite state systems [7, 9, 27]. Accordingly, we introduce 2-Partitions-Relation triples (2PR), generalizing the partition-relation pairs used in the most efficient symbolic simulation algorithms [7, 11, 30] as symbolic representation of a relation between states. We exploit here 2PRs to design a symbolic version of Algorithm 1. The rationale behind the need for 2PRs rather than partition-relation pairs, i.e. 1PR, has more to do with enhancing the presentation and ease of understanding and less to do with limitations of 1PRs.

Definition 6.1 (2PR Triple). Given an (infinite) set Σ, a triple $\langle P, \tau, Q\rangle$ with $P, Q \in \mathrm{Part}(\Sigma)$ and $\tau\colon P \to \wp(Q)$, is a *2-Partitions-Relation* (2PR) *triple*. $\diamond$

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 defined 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 $R_{\langle P,\tau,Q \rangle}(x) \triangleq \cup\tau(P(x))$ (namely, 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. It is routine to check that $P \preceq P_{\langle P,\tau,Q \rangle}$ where $P_{\langle P,\tau,Q \rangle} \in \mathrm{Part}(\Sigma)$ is induced by $R_{\langle P,\tau,Q \rangle}$.

We put forward Algorithm 2, designed as a refinement of Algorithm 1 representing R as a 2PR triple. Algorithm 2 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 which are candidate to simulate x, and the states in $P_{\langle P,\tau,Q \rangle}(x)$ are candidates to be simulation equivalent to x.

Algorithm 2: 2PR-based Algorithm

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

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

Following the idea of Algorithm 1, this symbolic procedure computes a set of principals $\{\cup\tau(B) \mid \cup\tau(B) \cap \sigma \neq \varnothing\}_{B \in P}$ 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 2 either updates reachability for the principal of some block in U by executing a *Search* iteration or stabilizes the pair of blocks associated to some triple in V by executing a *Refine* iteration. *Refine* iterations a-stabilize a pair of blocks (B, C) by possibly splitting B into $B \cap \mathsf{pre}_a(C)$ and $B \setminus \mathsf{pre}_a(C)$ (lines 13–15). Then, it refines the principal $\cup\tau(B \cap \mathsf{pre}_a(C))$ 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 $\mathsf{pre}_a(\cup\tau(C))$.

At termination, we have that $R^{\sigma}_{\langle P,\tau,Q\rangle}$ coincides with the set of reachable principals of R_{sim} (cf. (2.a) below), while we obtain an over-approximation of $P_{\mathrm{sim}}^{\mathsf{post}^*(I)}$ (cf. (2.b) below). Similarly to Algorithm 1, the term "over-approximation" is used w.r.t. the set $P_{\mathrm{sim}}^{\mathsf{post}^*(I)}$ itself, and not to the contained blocks, whose elements are computed in an exact way. The reader might find surprising the need of an $\exists$ quantifier in (2.b). We have that (2.b) provides a statement equivalent to (1.b), but since $P_{\langle P,\tau,Q\rangle}$ is, in general, coarser than P (as stated previously), then B is not guaranteed to be in the domain of τ, hence the existential quantifier.

Theorem 6.2 (Correctness of Algorithm 2 for Finite Systems). *Let $\langle P, \tau, Q, \sigma\rangle$ be the output of Algorithm 2 on input G with $|\Sigma| \in \mathbb{N}$, $R_i \in \mathrm{PreO}(\Sigma)$, and $\sigma_i \subseteq \mathsf{post}^*(I)$. Moreover, let $P_{\langle P,\tau,Q\rangle} \in \mathrm{Part}(\Sigma)$ be the partition induced by $R_{\langle P,\tau,Q\rangle}$. Then:*

$$R_{\mathrm{sim}}^{\mathsf{post}^*(I)} = R^{\sigma}_{\langle P,\tau,Q\rangle} \ , \tag{2.a}$$

$$P_{\mathrm{sim}}^{\mathsf{post}^*(I)} \subseteq \{B \in P_{\langle P,\tau,Q\rangle} \mid \exists E \in P.\ E \subseteq B \wedge (\cup\tau(E)) \cap \sigma \neq \varnothing\} \ . \tag{2.b}$$

As done for Algorithm 1 and Theorem 5.3, we extend Theorem 6.2 to infinite transition systems as follows.

Theorem 6.3 (Correctness of Algorithm 2). *Let $\langle P, \tau, Q, \sigma\rangle$ be the output of Algorithm 2 on input G, $R_i \in \mathrm{PreO}(\Sigma)$, and $\sigma_i \subseteq \mathsf{post}^*(I)$. 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 (2.a) and (2.b) are satisfied.*

Termination and Complexity of Algorithm 2. We provide two conditional termination results for Algorithm 2 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 execution. Our first conditional termination result is akin to that of Lee and Yannakakis [23, Th. 3.1 and the following paragraph therein], and applies when the bisimulation partition P_{bis} is finite. Moreover, it turns out that Algorithm 2 carries out a total number of iterations which is at most quadratic in the size of P_{bis}.

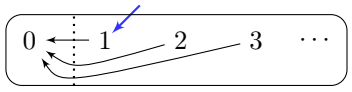
Theorem 6.4 (Termination of Algorithm 2). *Let G, $R_i \in \mathrm{PreO}(\Sigma)$ and $\sigma_i \subseteq \mathsf{post}^*(I)$ be the input of Algorithm 2. Moreover, let $P_{\mathrm{bis}} \in \mathrm{Part}(\Sigma)$ be the bisimulation partition w.r.t. P_i, the partition induced by R_i. If P_{bis} consists of finitely many blocks then the number of iterations of Algorithm 2 is in $O(|P_{\mathrm{bis}}|^2)$.*

A second termination result holds under a local finiteness condition: Algorithm 2 terminates if the number of blocks of P_{bis} contained in $R_i(\mathsf{post}^*(I))$ is finite. The result comes with a quadratic bound on the number of iterations.

Theorem 6.5 (Alternative Termination of Algorithm 2). *Let G, $R_i \in \mathrm{Part}(\Sigma)$ and $\sigma_i \subseteq \mathsf{post}^*(I)$ be the input of Algorithm 2 (note that R_i is a partition). Let $P_{\mathrm{bis}} \in \mathrm{Part}(\Sigma)$ be the bisimulation partition w.r.t. R_i, and let $T \triangleq \{Y \in P_{\mathrm{bis}} \mid R_i(Y) \cap \mathsf{post}^*(I) \neq \varnothing\}$ $(= \{Y \in P_{\mathrm{bis}} \mid Y \subseteq R_i(\mathsf{post}^*(I))\})$. If T is finite then the number of iterations of Algorithm 2 is in $O(|T|^2)$.*

It follows from Theorem 6.4 that Algorithm 2 terminates on all finite state systems (as does Algorithm 1). It turns out that these two termination results for Algorithm 2 are indeed stronger and extend to many infinite state systems as well. In fact, the use of 2PR triples as a representation structure in Algorithm 2 brings significant benefits in terms of termination on infinite systems, since there exist many inputs on which Algorithm 1 does not terminate, while Algorithm 2 does. The intuition here is that, through the use of 2PR triples, Algorithm 2 can refine infinitely many principals of the relation encoded by $\langle P, \tau, Q\rangle$, in a single *Refine* iteration. Below we show two examples which compare executions of Algorithms 1 and 2 on one input, illustrating this behaviour.

Example 6.6. Consider the following infinite transition system, where $\sigma_i = I = \{1\}$, and $R_i = \mathbb{N} \times \mathbb{N}$.



The simulation preorder R_{sim} is such that $R_{\mathrm{sim}}(0) = \mathbb{N}$, and $R_{\mathrm{sim}}(n) = \mathbb{N} \setminus \{0\}$ for $n \geq 1$, meaning that $P_{\mathrm{sim}} = \{\{0\}, \mathbb{N} \setminus \{0\}\}$. As usual, the block induced by R_i is represented as a box, and the dotted line delimits the blocks of P_{sim}. During execution of Algorithm 1, the set U is empty, since the state 1 is in 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 has infinitely many elements, and each iteration removes exactly one element from it, thus V is never empty. ⋄

On the other hand, Algorithm 2 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 6.6 and show that Algorithm 2 converges in a few iterations.

Example 6.7. Consider the input of Example 6.6. The initial 2PR is given by $P = Q = \{\mathbb{N}\}$ and $\tau(\mathbb{N}) = \{\mathbb{N}\}$, $U = \emptyset$, 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 2 terminates. ◇

7 Conclusion and Future Work

We introduced and proved the correctness and termination of algorithms solving the reachable simulation problem. We showed fundamental differences w.r.t. to the analogous problem for bisimulation studied by Lee and Yannakakis [23] in 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 2 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 2 offers the same termination guarantee as the LY algorithm [23]. On top of that, our algorithm also terminates under a local finiteness hypothesis, while LY [23] 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[7] represented systems gives rise to a multitude of questions which are domain dependent. Solving these are crucial to obtain an efficient implementation. Orthogonally, further generalizations of our algorithm—for instance, to other behavioral relations such as branching bisimilarity for labeled transition systems or stuttering equivalence for Kripke structures—constitute future research paths.

Acknowledgements. Francesco Ranzato was partially funded by: the *Italian MUR*, under the PRIN 2022 PNRR project no. P2022HXNSC; *Meta* (formerly *Facebook*) *Research*, under a "Probability and Programming Research Award" and under a *WhatsApp Research Award* on "Privacy-aware Program Analysis"; by an *Amazon Research Award* for "AWS Automated Reasoning". Nicolas Manini is supported by the grant PIPF-2022/COM-24370, funded by the Madrid Regional Government. This publication is part of the grant PID2022-138072OB-I00, funded by MCIN/AEI/10.13039/501100011033/ FEDER, UE and part of the PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR.

References

1. Alur, R., Henzinger, T.A.: Computer-Aided Verification (1999), chapter 4: Graph Minimization. (Unpublished manuscript)

[7] Reachability analysis is mostly superfluous on explicitly represented systems as they usually do not encode unreachable states.

2. Bensalem, S., Bouajjani, A., Loiseaux, C., Sifakis, J.: Property preserving simulations. In: Proc. of the Fourth International Workshop on Computer Aided Verification, CAV'92. pp. 260–273. LNCS, Springer (1992). https://doi.org/10.1007/3-540-56496-9_21
3. Bloom, B., Paige, R.: Transformational design and implementation of a new efficient solution to the ready simulation problem. Sci. Comput. Program. **24**(3), 189–220 (1995). https://doi.org/10.1016/0167-6423(95)00003-B
4. Bouajjani, A., Fernandez, J.C., Halbwachs, N.: Minimal model generation. In: Proc. of the International Workshop on Computer-Aided Verification, CAV'91. pp. 197–203. LNCS, Springer (1991). https://doi.org/10.1007/BFb0023733
5. Bouajjani, A., Fernandez, J.C., Halbwachs, N., Raymond, P., Ratel, C.: Minimal state graph generation. Science of Computer Programming **18**(3), 247–269 (1992). https://doi.org/10.1016/0167-6423(92)90018-7
6. Bustan, D., Grumberg, O.: Simulation-based minimization. ACM Trans. Comput. Log. **4**(2), 181–206 (2003). https://doi.org/10.1145/635499.635502
7. Cécé, G.: Foundation for a series of efficient simulation algorithms. In: Proc. of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017. pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005069
8. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R.: Handbook of Model Checking. Springer, 1st edn. (2018)
9. Crafa, S., Ranzato, F., Tapparo, F.: Saving space in a time efficient simulation algorithm. Fundam. Informaticae **108**(1-2), 23–42 (2011). https://doi.org/10.3233/FI-2011-412
10. Fisler, K., Vardi, M.Y.: Bisimulation minimization and symbolic model checking. Formal Methods Syst. Des. **21**(1), 39–78 (2002). https://doi.org/10.1023/A:1016091902809
11. Gentilini, R., Piazza, C., Policriti, A.: From bisimulation to simulation: Coarsest partition problems. Journal of Automated Reasoning **31**(1), 73–103 (2003). https://doi.org/10.1023/A:1027328830731
12. Glabbeek, R.v., Ploeger, B.: Correcting a space-efficient simulation algorithm. In: Proc. of the International Conference on Computer Aided Verification, CAV'08. pp. 517–529. Springer (2008). https://doi.org/10.1007/978-3-540-70545-1_49
13. Gratzer, G.A.: Lattice theory: foundation. Springer (2011)
14. Grumberg, O., Long, D.E.: Model checking and modular verification. In: Proc. of the 2nd International Conference on Concurrency Theory, CONCUR'91. pp. 250–265. LNCS, Springer (1991). https://doi.org/10.1007/3-540-54430-5_93
15. Grumberg, O., Long, D.E.: Model checking and modular verification. ACM Transactions on Programming Languages and Systems (TOPLAS) **16**(3), 843–871 (1994). https://doi.org/10.1145/177492.177725
16. Gulavani, B.S., Henzinger, T.A., Kannan, Y., Nori, A.V., Rajamani, S.K.: SYNERGY: a new algorithm for property checking. In: Proc. of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006. pp. 117–127. ACM (2006). https://doi.org/10.1145/1181775.1181790
17. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: Proc. of IEEE 36th Annual Foundations of Computer Science, FOCS'95. pp. 453–462 (1995). https://doi.org/10.1109/SFCS.1995.492576
18. Henzinger, T.A., Kopke, P.W.: Hybrid Automata with Finite Mutual Simulations. Tech. Rep. TR-95-1497, Computer Science Departement (1995)

19. Hofman, P., Lasota, S., Mayr, R., Totzke, P.: Simulation problems over one-counter nets. Logical Methods in Computer Science **12** (2016). https://doi.org/10.2168/LMCS-12(1:6)2016
20. Kučera, A., Jančar, P.: Equivalence-checking on infinite-state systems: Techniques and results. Theory and Practice of Logic Programming **6**(3), 227–264 (2006). https://doi.org/10.1017/S1471068406002651
21. Kucera, A., Mayr, R.: Simulation preorder over simple process algebras. Inf. Comput. **173**(2), 184–198 (2002). https://doi.org/10.1006/inco.2001.3122
22. Kucera, A., Mayr, R.: Why is simulation harder than bisimulation? In: Proc. of the 13th International Conference on Concurrency Theory, CONCUR 2002. pp. 594–610. LNCS, Springer (2002). https://doi.org/10.1007/3-540-45694-5_39
23. Lee, D., Yannakakis, M.: Online Minimization of Transition Systems. In: Proc. of the 24th Annual ACM Symposium on Theory of Computing, STOC '92. pp. 264–274. ACM (1992). https://doi.org/10.1145/129712.129738
24. Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., Bensalem, S.: Property preserving abstractions for the verification of concurrent systems. Formal Methods Syst. Des. **6**(1), 11–44 (1995). https://doi.org/10.1007/BF01384313
25. Majumdar, R., Ozay, N., Schmuck, A.K.: On abstraction-based controller design with output feedback. In: Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control, HSCC 2020. pp. 1–11. ACM (2020). https://doi.org/10.1145/3365365.3382219
26. Pasareanu, C.S., Pelánek, R., Visser, W.: Concrete model checking with abstract matching and refinement. In: Proc. 17th International Conference on Computer Aided Verification, CAV 2005. pp. 52–66. LNCS, Springer (2005). https://doi.org/10.1007/11513988_7
27. Ranzato, F.: A more efficient simulation algorithm on Kripke structures. In: Proc. of the 38th International Symposium on Mathematical Foundations of Computer Science 2013, MFCS 2013. pp. 753–764. LNCS, Springer (2013). https://doi.org/10.1007/978-3-642-40313-2_66
28. Ranzato, F.: An efficient simulation algorithm on Kripke structures. Acta informatica **51**(2), 107–125 (2014). https://doi.org/10.1007/s00236-014-0195-9
29. Ranzato, F., Tapparo, F.: A new efficient simulation equivalence algorithm. In: Proc. of the 22nd IEEE Symposium on Logic in Computer Science, LICS 2007. pp. 171–180. IEEE Computer Society (2007). https://doi.org/10.1109/LICS.2007.8
30. Ranzato, F., Tapparo, F.: An efficient simulation algorithm based on abstract interpretation. Information and Computation **208**(1), 1–22 (2010). https://doi.org/10.1016/j.ic.2009.06.002
31. Tan, L., Cleaveland, R.: Simulation revisited. In: Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001. pp. 480–495. LNCS, Springer (2001). https://doi.org/10.1007/3-540-45319-9_33
32. van Glabbeek, R., Ploeger, B.: Five determinisation algorithms. In: Implementation and Applications of Automata. pp. 161–170. LNCS, Springer (2008). https://doi.org/10.1007/978-3-540-70844-5_17
33. Yannakakis, M., Lee, D.: An efficient algorithm for minimizing real-time transition systems: Extended abstract. Formal Methods in System Design **11**(2), 113–136 (1997). https://doi.org/10.1023/A:1008621829508