

# Computing Reachable Simulations

Pierre Ganty  
IMDEA Software Institute  
Pozuelo de Alarcón, Spain  
pierre.ganty@imdea.org

Nicolas Manini  
IMDEA Software Institute and  
Universidad Politécnica de Madrid  
Pozuelo de Alarcón, Spain  
nicolas.manini@imdea.org

Francesco Ranzato  
University of Padova  
Padova, Italy  
francesco.ranzato@unipd.it

**Abstract**—We study the problem of computing the reachable blocks of simulation equivalence and design algorithms for this problem by interleaving reachability and simulation computation while possibly avoiding the computation of all the reachable states or the whole simulation preorder. Following an investigation of the decidability and complexity aspects of this problem, we put forward several sound algorithms with varying completeness guarantees as well as a symbolic procedure manipulating state partitions and relations between their blocks, suited for processing infinite state systems. We show that the symbolic algorithm comes with better termination guarantees and, through empirical evidence, runs faster in general.

## I. INTRODUCTION

Given a possibly infinite labeled transition system, we study the problem of computing the reachable blocks of the partition (equivalently, equivalence) induced by the simulation preorder, i.e. the greatest simulation relation, here called *reachable* (part of) *simulation*. By reachable we mean the blocks of the partition that intersect the reachable states of the system, therefore ignoring unreachable blocks which, typically, are of no/negligible interest.

Two possible use cases of computing the reachable blocks of the simulation partition include transition system minimization and efficient determinization. In the former, the reachable blocks of simulation partition define the states of the reduced system. The question of computing the transitions between the blocks of the reduced system has already been investigated in depth by Bustan and Grumberg [1] who explore the difference and trade-offs of the  $\exists\exists$  (i.e.,  $B \rightarrow B'$  iff  $\exists s \in B. \exists s' \in B'. s \rightarrow s'$ ) and  $\forall\exists$  definitions (i.e.,  $B \rightarrow B'$  iff  $\forall s \in B. \exists s' \in B'. s \rightarrow s'$ ) for transitions between blocks and designed an algorithm for computing the minimal number of transitions. For efficient determinization, the simulation relation is leveraged during the determinization to compute a deterministic automata that could have been larger had the simulation relation not been used [2]. Since the determinization is carried forward starting from the initial states, the information provided by the unreachable blocks of the simulation equivalence (i.e. those containing no reachable states) is useless.

A naïve solution to this reachable simulation problem would be: first, compute the simulation partition, and then, filter out the blocks containing no reachable states. This, however, would require the computation of the entire simulation preorder and of, possibly, all the reachable states. In this

work, we will present a completely different solution relying on a **convoluted interleaving of reachability and simulation computation** that allows us to possibly avoid the computation of all the reachable states and of the whole simulation preorder.

## Contributions

In Section III, we show, by means of a reduction to an undecidable termination problem for infinite state systems, and then, for finite state systems, through a reduction to the st-connectivity NL-complete problem for directed graphs, that there is stark contrast between the problem of **computing the reachable blocks of the bisimulation partition, settled in 1992 by Lee and Yannakakis [3]**, and the analogous problem for the simulation partition tackled by this paper. In Section IV, we put forward a first algorithm which solves the reachable simulation problem in a complete way for the simulation preorder while yielding a sound over-approximation for simulation equivalence. Next, we design in Section V a second algorithm which calls the previous algorithm as a subroutine and turns out to be complete with respect to simulation equivalence. Our results prove correctness and termination of these two algorithms for finite state systems and discuss how correctness is achieved under simple assumptions for the case of infinite state systems as well. Moreover, we provide examples showing correctness and termination on infinite inputs. Section VII introduces the notion of 2PR triple consisting of two state partitions (2P) and of a relation (R) between the blocks of these two partitions, which allows us to design a symbolic version of the algorithm in Section IV, where a relation between states is symbolically represented through a 2PR. Besides inheriting the termination and correctness guarantees of the above algorithms, the 2PR based algorithm terminates faster and more often for the case of infinite state systems. Finally, in Section VIII we report on a prototype implementation of our algorithms which is run on a set of benchmarks arising from some mutual exclusion protocols such as Fischer's and Peterson's algorithms. These experiments provide an empirical evidence that the symbolic algorithm runs much faster than an explicit implementation of the first algorithm.

## Related Work

The closest work to ours is that by Lee and Yannakakis [3], who first designed in STOC 1992 an intricate interleaving of reachability and bisimulation computation, here referred to as

the LY algorithm. To put the LY algorithm in its historical context, it was one of several algorithms to compute the reachable part of the bisimulation quotiented system [3]–[5]. These algorithms have in common the interleaving of the bisimulation computation—which itself is a partition refinement algorithm—with the computation determining which bisimulation blocks are reachable. The remarkable interest of interleaving reachability and bisimulation computation is that the resulting algorithms terminate as least as often (possibly more often) than the naïve procedure consisting in first computing the bisimulation and next determining its reachable blocks. Later on, these algorithms for computing the reachable bisimulation have been revisited by Alur and Henzinger in a chapter of their unpublished book on computer-aided verification [6, Chapter 4], as well as by the theoretical and experimental comparison made by Fisler and Vardi [7]. Let us 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 [8], [9] to hybrid systems verification [10].

Our focus on simulation stems from the well-known fact that it provides a better state space reduction than bisimilarity, yet the similarity quotient retains enough precision for checking all linear temporal formulas or branching temporal logic formulas without quantifier switches [11]–[15]. Moreover, infinite state systems like 2D rectangular automata may have infinite bisimilarity quotients, yet their similarity quotients are always finite [16].

There is a large body of work [16]–[26] on efficiently computing the simulation preorder, leveraging both explicit or symbolic algorithms. One major interest for simulation algorithms comes from the fact that simulation-based minimization of (labeled) transition systems strongly preserves  $\forall\text{CTL}^*$  formulas [1], [13], [14]. Kučera and Mayr [27], [28] have compared simulation and bisimulation equivalence from the perspective of the computational complexity for deciding them, and gave precise justifications to 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 blocks of the partition induced by the simulation preorder.

Due to lack of space, most proofs and some additional material are deferred as appendices.

## II. BACKGROUND

*a) Orders and Partitions:* Given a (possibly infinite) set  $\Sigma$ , we denote with  $\wp(\Sigma)$  the powerset of  $\Sigma$ , and with  $\text{Rel}(\Sigma) \triangleq \wp(\Sigma \times \Sigma)$  the set of relations over  $\Sigma$ . If  $R \in \text{Rel}(\Sigma)$  then: for  $S \in \wp(\Sigma)$ ,  $R(S) \triangleq \{s' \in \Sigma \mid \exists s \in S. (s, s') \in R\}$ ; for  $s \in \Sigma$ , the set  $R(s) \triangleq R(\{s\})$  is the *principal* of  $s$ ;  $\text{Rel}(\Sigma) \ni R^{-1} \triangleq \{(y, x) \in \Sigma \times \Sigma \mid (x, y) \in R\}$  is the *converse* relation of  $R$ . Moreover, for a given set  $S \in \wp(\Sigma)$  we denote  $R^S \triangleq \{R(x) \in \wp(\Sigma) \mid x \in \Sigma, R(x) \cap S \neq \emptyset\}$ . A relation  $R \in \text{Rel}(\Sigma)$  is a *quasiorder* (qo, sometimes also called preorder) if it is reflexive and transitive, and  $\text{QO}(\Sigma) \triangleq$

$\{R \in \text{Rel}(\Sigma) \mid R \text{ is a qo}\}$  denotes the set of quasiorders on  $\Sigma$ . Analogously,  $R \in \text{Rel}(\Sigma)$  is an *equivalence* if it is a symmetric qo. A *partition* of  $\Sigma$  consists of pairwise disjoint nonempty subsets of  $\Sigma$ , called *blocks*, whose union is  $\Sigma$ . An equivalence relation in  $\text{Rel}(\Sigma)$  induces a partition of  $\Sigma$  where each block is an equivalence class, and vice versa. Define  $\text{Part}(\Sigma)$  to be the set of partitions of  $\Sigma$ . Given a partition  $P \in \text{Part}(\Sigma)$ ,  $P(s)$ ,  $P(S)$  and  $P^S$  (for  $S \in \wp(\Sigma), s \in \Sigma$ ) are well defined thanks to the equivalence underlying  $P$ . In particular,  $P(s)$  is the block including  $s$ ,  $P(S) = \cup\{P(s) \in P \mid s \in S\}$ , and  $P^S = \{P(s) \in P \mid s \in S\} \in \text{Part}(P(S))$ . Given two partitions  $P, Q \in \text{Part}(\Sigma)$ , we denote with  $P \wedge Q \in \text{Part}(\Sigma)$  the partition obtained by intersecting the underlying equivalence relations (this coincides with the meet in the lattice of partitions w.r.t. their “finer than” partial order).

*b) Simulation and Bisimulation:* Let  $G = (\Sigma, I, L, \rightarrow)$  be a (labeled) transition system, where  $\Sigma$  is a (possibly infinite) set of states,  $I \subseteq \Sigma$  is a subset of *initial states*,  $L$  is a finite set of action labels, and  $\rightarrow \in \wp(\Sigma \times L \times \Sigma)$  is the *transition relation*, where we denote  $(x, a, y) \in \rightarrow$  as  $x \xrightarrow{a} y$ , and  $x \rightarrow y \triangleq \exists a \in L. x \xrightarrow{a} y$ . When the set  $L$  is a singleton, we leverage the previous notation when writing  $x \rightarrow y$ , and equivalently we also say that  $(x, y) \in \rightarrow$  when no ambiguity arises. Given  $a \in L$ ,  $\text{post}_a: \wp(\Sigma) \rightarrow \wp(\Sigma)$  denotes the usual *successor transformer*  $\text{post}_a(X) \triangleq \{y \in \Sigma \mid \exists x \in X. x \xrightarrow{a} y\}$ , and, dually,  $\text{pre}_a: \wp(\Sigma) \rightarrow \wp(\Sigma)$  is the *predecessor*  $\text{pre}_a(X) \triangleq \{y \in \Sigma \mid \exists x \in X. y \xrightarrow{a} x\}$ . Moreover, we define  $\text{post}: \wp(\Sigma) \rightarrow \wp(\Sigma)$  as  $\text{post}(X) \triangleq \cup_{a \in L} \text{post}_a(X)$  and, symmetrically,  $\text{pre}: \wp(\Sigma) \rightarrow \wp(\Sigma)$  as  $\text{pre}(X) \triangleq \cup_{a \in L} \text{pre}_a(X)$ . Thus,  $\text{post}^*(I) = \cup_{n \in \mathbb{N}} \text{post}^n(I)$  is the set of *reachable states* (from the initial states) of  $G$ .

Given an (initial) qo  $R_i \in \text{QO}(\Sigma)$  (e.g.,  $R_i$  can be the partition induced by the initial states or by some labeling of a Kripke structure), a relation  $R \in \text{Rel}(\Sigma)$  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  $R(s) \subseteq \text{pre}_a(R(s'))$ , otherwise  $R(s)$  is called *a-unstable* (or just *unstable*) w.r.t.  $R(s')$ , and, in this case,  $R(s')$  can *refine*  $R(s)$ . The greatest (w.r.t.  $\subseteq$ ) simulation relation on  $G$  exists and turns out to be a qo called *simulation quasiorder* of  $G$  w.r.t.  $R_i$ , denoted by  $R_{\text{sim}} \in \text{QO}(\Sigma)$ , while  $P_{\text{sim}} \in \text{Part}(\Sigma)$  is the *simulation partition* induced by the *similarity* equivalence  $R_{\text{sim}} \cap (R_{\text{sim}})^{-1}$ . A relation  $R \in \text{Rel}(\Sigma)$  is a *bisimulation* on  $G$  w.r.t. an (initial) partition  $P_i \in \text{Part}(\Sigma)$  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), whose corresponding *bisimulation partition* is denoted by  $P_{\text{bis}} \in \text{Part}(\Sigma)$ .

## III. THE PROBLEM OF COMPUTING REACHABLE SIMULATIONS

This work addresses the problem of computing the reachable blocks of simulation equivalence by interleaving reachability and simulation computation.

### Problem III.1 (Reachable Simulation Problem).

GIVEN: An effectively presented<sup>1</sup> labeled transition system  $G = (\Sigma, I, L, \rightarrow)$  and an initial qo  $R_i \in \text{QO}(\Sigma)$ .

COMPUTE: The reachable principals of  $R_{\text{sim}}$  and the reachable blocks of  $P_{\text{sim}}$ , where  $R_{\text{sim}}$  and  $P_{\text{sim}}$  are, resp., the simulation preorder and partition on  $G$  w.r.t.  $R_i$ . ♦

One first challenge we face in this problem is related to the notion of reachability. Let us observe that the notion of reachability for blocks of a partition  $P \in \text{Part}(\Sigma)$ —such as the partition  $P_{\text{sim}}$  induced by simulation equivalence—is naturally defined as follows:

$$\begin{aligned} P^{\text{post}^*(I)} &= \{B \in P \mid B \cap \text{post}^*(I) \neq \emptyset\} \\ &= \{P(s) \in P \mid s \in \text{post}^*(I)\}. \end{aligned} \quad (1)$$

When considering simulations rather than bisimulations, a notion of reachability for principals of a relation is also needed, leading to multiple generalizations of the above notion of reachability for blocks (1). In fact, reachability for principals is not uniquely formulated, and as such, the two following different definitions of *reachable principal* of a reflexive relation  $R \in \text{Rel}(\Sigma)$  can both be considered adequate:

$$\begin{aligned} \{R(s) \in \wp(\Sigma) \mid s \in \Sigma, R(s) \cap \text{post}^*(I) \neq \emptyset\}, \quad (2) \\ \{R(s) \in \wp(\Sigma) \mid s \in \text{post}^*(I)\}. \quad (3) \end{aligned}$$

Here, clearly, (3)  $\subseteq$  (2) holds, and the following example shows that this inclusion may be strict:

**Example III.2.** Let us consider the transition system  $(\Sigma = \{0, 1\}, I = \{1\}, L = \{a\}, \rightarrow = \{(1, 1)\})$  and the initial qo  $R_i = \Sigma \times \Sigma$ . The simulation qo induced by  $R_i$  is given by  $R_{\text{sim}}(0) = \{0, 1\}$ ,  $R_{\text{sim}}(1) = \{1\}$ . Therefore, since  $\text{post}^*(I) = \{1\}$ , the only reachable principal according to (3) is  $R_{\text{sim}}(1)$ , while the reachable principals for (2) are both  $R_{\text{sim}}(0)$  and  $R_{\text{sim}}(1)$ . ♦

Let us remark that, for some systems, the reachable principals according to (2) could be infinitely many, while for (3) could be finitely many.

#### A. Undecidability for Infinite State Systems

We show that the problem of computing the reachable blocks in  $P_{\text{sim}}^{\text{post}^*(I)}$  of a simulation partition  $P_{\text{sim}}$  over an infinite transition system is, in general, unsolvable even under the assumption that  $P_{\text{sim}}$  is a finite partition. This impossibility result is in stark contrast with the problem of computing reachable blocks for the bisimulation partition  $P_{\text{bis}}$  which has been shown solvable by Lee and Yannakakis [3, Theorem 3.1 and the following paragraph]. To prove our impossibility result, we show that the undecidable halting problem for 2-counter machines can be reduced to the reachability problem for the finitely many blocks of a simulation partition.

Given  $n \geq 1$ , a counter machine  $n$ -CM is a tuple  $M = (Q, C, \Delta, q_0, H)$ , where:  $Q$  is a finite set of states including an initial state  $q_0$ ,  $C$  is a finite set of  $n$  counter variables storing

natural numbers,  $H \subseteq Q$  is a set of halting states disjoint from  $q_0$ , and  $\Delta \subseteq Q \times Q$  is a set of transitions of three types:

- 1)  $q \xrightarrow{c:=c+1} q'$ , which increases the counter  $c$ ;
- 2)  $q \xrightarrow{c:=c-1} q'$ , which decreases the counter  $c$ : this type of transitions can only be taken if the counter  $c$  stores a positive value;
- 3)  $q \xrightarrow{c=0} q'$ , which performs a zero-test: these transitions can only be taken if the counter  $c$  stores the value 0.

A configuration of  $M = (Q, C, \Delta, q_0, H)$  is a tuple  $(q, j_1, \dots, j_{|C|}) \in Q \times \mathbb{N}^{|C|}$ , where  $q \in Q$  is a state and  $j_1, \dots, j_{|C|} \in \mathbb{N}$  are the values stored by the  $|C|$  counters of  $M$ . A configuration  $(q, j_1, \dots, j_{|C|})$  is halting when  $q \in H$ . The operational semantics of a counter machine is determined by a transition relation  $\rightarrow$  between configurations and defined as expected. A counter machine  $M$  halts on input  $(j_1, \dots, j_{|C|})$  if there exist configurations  $c_1, \dots, c_k$  such that:  $c_1 \rightarrow \dots \rightarrow c_k$ ;  $c_1 = (q_0, j_1, \dots, j_{|C|})$ ;  $c_k$  is halting. The halting problem for 2-CMs is known to be undecidable [29]:

### Problem III.3 (Halting of 2-CMs).

GIVEN: A 2-CM  $M$ .

DECIDE: Can  $M$  halt on input  $(0, 0)$ ? ♦

We will show by reduction that since Problem III.3 is undecidable then so must be the following problem:

### Problem III.4 (Reachability for Simulation Partitions).

GIVEN: A 2-CM  $M = (Q, C, \Delta, q_0, H)$  and an effectively presented finite simulation partition  $P_{\text{sim}}$  of the transition system  $(\Sigma = Q \times \mathbb{N}^2, I = \{(q_0, 0, 0)\}, L = \{a\}, \rightarrow)$  generated by  $M$  w.r.t. an initial qo relation on  $Q \times \mathbb{N}^2$ .

DECIDE:  $\forall B \in P_{\text{sim}}: \text{post}^*(\{(q_0, 0, 0)\}) \cap B \neq \emptyset$ ? ♦

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  $q_c$  for each non-halting state  $q$  of  $M$ . Besides the transitions of  $M$ ,  $M_1$  has two additional transitions for each new state  $q_c$  added at the previous step:  $q \xrightarrow{c:=c+1} q_c$  and  $q_c \xrightarrow{c:=c-1} q$ , where  $c$  is one of the two counters of  $M$ . 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 an halting state  $h \in H$  has outgoing transitions then we define  $M_2$  by duplicating in  $M_1$  the state  $h$  and all its incident 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  $I \triangleq \{(q_0, 0, 0)\}$ .

<sup>1</sup>There is a finite encoding on the tape of the underlying Turing machine.

By considering as initial qo relation

$$R_i \triangleq ((H \times \mathbb{N}^2) \times (Q \times \mathbb{N}^2)) \cup (((Q \setminus H) \times \mathbb{N}^2) \times ((Q \setminus H) \times \mathbb{N}^2)) \in \text{QO}(Q \times \mathbb{N}^2)$$

we show that  $R_i$  is the simulation preorder, i.e.  $R_i = R_{\text{sim}}$ , because we can prove that for every transition  $c_x \rightarrow c_y$  the inclusion  $R_i(c_x) \subseteq \text{pre}(R_i(c_y))$  holds. Firstly, note that if  $c_x$  is halting then  $c_x \rightarrow c_y$  for no configuration  $c_y$  (the definition of  $M_2$  guarantees this property). If  $c_x$  is non-halting then we have that  $R_i(c_x) = (Q \setminus H) \times \mathbb{N}^2$ . If  $c_y$  is halting then we have  $R_i(c_y) = Q \times \mathbb{N}^2$ , 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  $R_i(c_x) \subseteq \text{pre}(R_i(c_y))$  holds. The last case to consider is when both  $c_x$  and  $c_y$  are non-halting, which implies that  $R_i(c_x) = R_i(c_y) = (Q \setminus H) \times \mathbb{N}^2$ . Here, we find that the inclusion  $R_i(c_x) \subseteq \text{pre}(R_i(c_y))$  holds since every element of  $(Q \setminus H) \times \mathbb{N}^2$  has an outgoing transition into some non-halting configuration.

Therefore  $R_i = R_{\text{sim}}$ , hence the simulation equivalence  $P_{\text{sim}}$ , defined as  $R_{\text{sim}} \cap (R_{\text{sim}})^{-1}$ , is given by:

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

Finally, it turns out that  $\{(q_0, 0, 0)\}$  can reach a halting configuration in  $M$  iff  $\text{post}^*(\{(q_0, 0, 0)\}) \cap (H \times \mathbb{N}^2) \neq \emptyset \wedge \text{post}^*(\{(q_0, 0, 0)\}) \cap ((Q \setminus H) \times \mathbb{N}^2) \neq \emptyset$  iff  $\forall B \in P_{\text{sim}}: \text{post}^*(\{(q_0, 0, 0)\}) \cap B \neq \emptyset$ . We have therefore shown the following result.

**Theorem III.5.** *Problem III.4 is undecidable.*

As a consequence, there exists no algorithm that, for an effectively presented transition system  $G$  and initial quasiorder  $R_i$ , is able to compute the reachable blocks of the simulation partition  $P_{\text{sim}}$  of  $G$  w.r.t.  $R_i$ , namely, the subset of blocks  $\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\}$ , even under the hypothesis that  $P_{\text{sim}}$  consists of a finite set of blocks. This negative result is to be contrasted with the positive result of Lee and Yannakakis for bisimulation equivalence and which states that the LY algorithm terminates when  $P_{\text{bis}}$  is finite [3, Theorem 3.1 and the following paragraph].

### B. Complexity for Finite State Systems

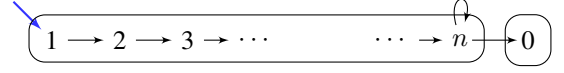
We show a further key difference between the problems of deciding reachability of blocks of  $P_{\text{sim}}$  and  $P_{\text{bis}}$  over *finite* transition systems. These two problems are obviously both decidable, since we can simply compute  $\text{post}^*(I)$ , and, then, check that whether  $\text{post}^*(I) \cap B = \emptyset$  holds for every block  $B$  in  $P_{\text{bis}}$  or  $P_{\text{sim}}$ .

For the case of bisimulation, deciding the reachability of  $B \in P_{\text{bis}}$  is solved in  $O(|P_{\text{bis}}^{\text{post}^*(I)}|)$  time by leveraging the definition of bisimulation: picking a pair of bisimilar states  $x$  and  $y$ , i.e. such that  $P_{\text{bis}}(x) = P_{\text{bis}}(y)$ , we have that  $\{B \in P_{\text{bis}} \mid \text{post}(x) \cap B \neq \emptyset\} = \{B \in P_{\text{bis}} \mid \text{post}(y) \cap B \neq \emptyset\}$ , namely, for  $B \in P_{\text{bis}}$ ,  $x$  can reach  $B$  iff  $y$  can reach  $B$ , hence picking one state per block of  $P_{\text{bis}}$  suffices.

For the simulation case, deciding the reachability of  $B \in P_{\text{sim}}$  is more involved than in the bisimulation case. As the

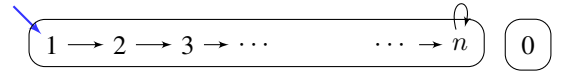
following example suggests, to decide whether  $B \in P_{\text{sim}}$  is reachable it seems unavoidable to have to decide whether there is a path of arbitrary length in the transition system reaching  $B$ .

**Example III.6.** Consider the following transition system  $G_1$  where  $I = \{1\}$  and  $L$  is a singleton (so the label is omitted).



Here, we have that  $P_{\text{sim}}$  w.r.t.  $R_i = \Sigma \times \Sigma$  is given by the two boxes depicted in the figure because  $R_{\text{sim}}(0) = [0, n]$  and, for all  $k \in [1, n]$ ,  $R_{\text{sim}}(k) = [1, n]$ , so that  $P_{\text{sim}} = \{[0, 0], [1, n]\}$  and, in turn, the block  $[0, 0] \in P_{\text{sim}}$  is actually reachable.

Let us remove the transition  $n \rightarrow 0$ , yielding the following system  $G_2$ :



In this case,  $R_{\text{sim}}$  and, therefore,  $P_{\text{sim}}$  remain the same as above, however the block  $[0, 0] \in P_{\text{sim}}$  becomes unreachable. Hence, in order to distinguish whether  $[0, 0]$  is reachable or not in these two instances, we have to detect that in  $G_1$  the state 0 is reachable, while in  $G_2$  it is not.

Let us observe that this is not the case for bisimulation, because we have that  $P_{\text{bis}} = \{[0, 0], [1, 1], [2, 2], \dots, [n, n]\}$  for  $G_1$ , while  $P_{\text{bis}} = P_{\text{sim}}$  for  $G_2$ . ♦

Let us now turn Example III.6 into a formal complexity argument by showing that, for finite transition systems with a two block (bi)simulation equivalence (this same argument of course applies to any fixed and finite number of blocks), deciding whether both blocks are reachable is NL-hard for the simulation equivalence while it is in L for the bisimulation equivalence. Formally stated the problem is as follows.

**Problem III.7 (Reachable Simulation Decision Problem for Finite Systems).**

GIVEN: A finite transition system  $G = (\Sigma, I, L, \rightarrow)$  and a two block simulation partition  $P_{\text{sim}} = \{B_0, B_1\}$  of  $G$ .

DECIDE:  $\text{post}^*(I) \cap B_0 \neq \emptyset$  and  $\text{post}^*(I) \cap B_1 \neq \emptyset$ ? ♦

**Theorem III.8.** *Problem III.7 is NL-hard, while the analogous decision problem for bisimulation partition is in L.*

The NL-hardness proof reduces from the st-connectivity problem in leveled directed graphs, while the L membership proof crucially leverages the definition of bisimulation. Intuitively, the situation described in Example III.6 cannot occur anymore for the case of bisimulation because its definition mandates that if a state of the block  $[1, n]$  has a transition to a state in  $[0]$  then each state of  $[1, n]$  has to have a transition to  $[0]$ . Roughly speaking, it turns out that if there is a path of arbitrary length then there is path of length one. Detailed proofs are given in Appendix A.



---

**Algorithm 1: Sound Algorithm**


---

**Input:** A transition system  $G = (\Sigma, I, L, \rightarrow)$ , an initial qo  $R_i \in \text{QO}(\Sigma)$ , an initial set  $\sigma_i \subseteq \text{post}^*(I)$ .

```

1 Rel( $\Sigma$ )  $\ni R := R_i$ ;
2  $\wp(\Sigma) \ni \sigma := \sigma_i$ ;
3 while true do
  // INV1:  $\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x)$ 
  // INV2:  $\sigma_i \subseteq \sigma \subseteq \text{post}^*(I)$ 
  // INV3:  $\forall x \in \Sigma. x \in R(x)$ 
4  $U := \{R(x) \mid R(x) \cap \sigma = \emptyset, R(x) \cap (I \cup \text{post}(\sigma)) \neq \emptyset\}$ ;
5  $V := \{\langle a, x, x' \rangle \in L \times \Sigma^2 \mid R(x) \cap \sigma \neq \emptyset, x \xrightarrow{a} x', R(x) \not\subseteq \text{pre}_a(R(x'))\}$ ;
6 nif
7    $(U \neq \emptyset) \longrightarrow \text{Search :}$ 
8     choose  $R(x) \in U$ ;
9     choose  $s \in R(x) \cap (I \cup \text{post}(\sigma))$ ;
10     $\sigma := \sigma \cup \{s\}$ ;
11    $(V \neq \emptyset) \longrightarrow \text{Refine :}$ 
12     choose  $\langle a, x, x' \rangle \in V$ ;
13      $R(x) := R(x) \cap \text{pre}_a(R(x'))$ ;
14    $(U = \emptyset \wedge V = \emptyset) \longrightarrow \text{return } \langle R, \sigma \rangle$ ;
15 fin
16 end while

```

---

#### IV. A SOUND REACHABLE SIMULATION ALGORITHM

We put forward Algorithm 1 which, given a transition system  $G$ , an initial qo  $R_i$  and an initial set of reachable nodes  $\sigma_i$  ( $\sigma_i$  can be empty), computes the reachable principals of  $R_{\text{sim}}$  according to definition (2), along with a sound overapproximation of the blocks of  $P_{\text{sim}}^{\text{post}^*(I)}$ .

This algorithm maintains a current relation  $R \in \text{Rel}(\Sigma)$  specified through its principals  $R(x) \in \wp(\Sigma)$ , and a set  $\sigma \in \wp(\Sigma)$  containing states reachable from  $I$ , so that  $R^\sigma \triangleq \{R(x) \mid R(x) \cap \sigma \neq \emptyset\}$  includes the principals which are currently known to be reachable, i.e., containing a reachable state. The algorithm computes the set  $U$  of principals that can be added to  $R^\sigma$  and the set  $V$  of unstable pairs of principals. A principal  $R(x)$  belongs to  $U$  if it contains an initial state or a successor of a state already known to be reachable. A triple  $\langle a, x, x' \rangle$  belongs to  $V$  if  $R(x)$  is known to be reachable and it can be refined by  $R(x')$ , 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 state relation  $R$ , the set of states  $\sigma$ , and the sets  $U$  and  $V$ .

Algorithm 1 either updates the reachability information for some principal in  $U$  or stabilizes the pair of principals associated to  $\langle a, x, x' \rangle \in V$  by refining  $R(x)$ . In our pseudocode we use a **nif** statement which is a *nondeterministic choice* between guarded commands. In that statement we have three guarded commands: either the *Search* (lines 7–10) or the *Refine* procedures (lines 11–13) are executed, or, at line 14,

when both their guards are false, the return statement is taken. Thus, every execution consists of an arbitrary interleaving of *Search* and *Refine*, possibly followed by the return at line 14. 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 13 only if it is known to be currently reachable. Upon termination,  $R^\sigma$  turns out to be the set of reachable principals of  $R_{\text{sim}}$  (cf. (1.a) below). However,  $R$  may well contain unstable principals, so that, in general,  $R$  and  $R_{\text{sim}}$  do not coincide. Turning to the simulation partition, Algorithm 1 yields a sound overapproximation of  $P_{\text{sim}}^{\text{post}^*(I)}$  (cf. (1.b) below).

**Theorem IV.1 (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  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$  and  $\sigma_i \subseteq \text{post}^*(I)$ . Let  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation qo and partition w.r.t.  $R_i$ . Let  $P \triangleq \{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma} \in \text{Part}(\Sigma)$ . Then:*

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

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

The reader might find surprising that (1.b) is not  $P_{\text{sim}}^{\text{post}^*(I)} \subseteq P^\sigma$ . Indeed  $P_{\text{sim}}^{\text{post}^*(I)} \subseteq P^\sigma$  does not hold in general as shown in the following example:

**Example IV.2.** Let us consider the system  $(\Sigma = \{1, 2\}, I = \{1\}, L, \rightarrow = \{(1, 2)\})$ , where  $L$  is a singleton. Moreover, fix  $\sigma_i = \emptyset$  and  $R_i = \{(1, 1), (2, 1), (2, 2)\} = R_{\text{sim}}$ . Here, we have that  $\text{post}^*(I) = \{1, 2\}$ , and  $P_{\text{sim}} = \{\{1\}, \{2\}\}$ . Algorithm 1 on input  $R_i, \sigma_i$  returns  $R = R_{\text{sim}}, \sigma = \{1\}$ . It follows that  $P_{\text{sim}}^{\text{post}^*(I)} = P_{\text{sim}}, P^\sigma = \{\{1\}\}$ , thus  $P_{\text{sim}}^{\text{post}^*(I)} \not\subseteq P^\sigma$ . Since  $\{B \in P \mid R(B) \cap \sigma \neq \emptyset\} = P_{\text{sim}}$  we have that (1.b) holds for this example. ♦

Moreover, the following example shows that the containment of (1.b) may be strict.

**Example IV.3.** Let us consider the system  $(\Sigma = \{1, 2\}, I = \{1\}, L, \rightarrow = \{(1, 1)\})$ , where  $L$  is a singleton. Moreover, fix  $\sigma_i = \emptyset$  and  $R_i = \{1, 2\} \times \{1, 2\}$ . Here, we have that  $\text{post}^*(I) = \{1\}$ ,  $R_{\text{sim}}(1) = \{1\}$ ,  $R_{\text{sim}}(2) = \{1, 2\}$ , so that  $P_{\text{sim}} = \{[1, 1], [2, 2]\}$ . Algorithm 1 on input  $R_i, \sigma_i$  returns  $R = R_{\text{sim}}, \sigma = \{1\}$ . Thus, the containment (1.b) turns out to be strict since  $\{[1, 1]\} \subsetneq \{[1, 1], [2, 2]\}$ . ♦

It turns out that Algorithm 1 always terminates on finite state systems.

**Theorem IV.4 (Termination of Algorithm 1).** *Let  $G = (\Sigma, I, L, \rightarrow)$  with  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$  and  $\sigma_i \subseteq \text{post}^*(I)$ . Then, Algorithm 1 terminates on input  $G, R_i$ , and  $\sigma_i$ .*

While correctness and termination are guaranteed for finite state systems, Algorithm 1 may well be correct and terminating on some infinite state systems, as shown in the next example. Let us also remark that in Appendix C we discuss how the proof of Theorem IV.1 can be extended to infinite systems.

**Algorithm 2: Sound Refined Algorithm (SymRef)**

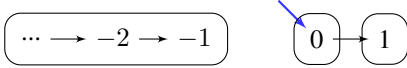
**Input:** A transition system  $G = (\Sigma, I, L, \rightarrow)$ , the set of reachable states  $\text{post}^*(I)$ , and a relation  $R_i \in \text{Rel}(\Sigma)$  s.t.  $R_{\text{sim}} \subseteq R_i \subseteq R_{\text{qo}}$ , where  $R_{\text{qo}}$  is some qo and  $R_{\text{sim}}$  is the simulation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

```

1 Rel( $\Sigma$ )  $\ni R := R_i$ ;
2  $\wp(\Sigma) \ni \sigma := \text{post}^*(I)$ ;
3 while true do
  // INV1:  $\forall x. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x)$ 
  // INV2:  $\forall x \in \Sigma. x \in R(x)$ 
4   $V := \{(a, x, x') \in L \times \Sigma^2 \mid R(x) \cap \sigma \neq \emptyset, x \xrightarrow{a} x', R(x) \not\subseteq \text{pre}_a(R(x'))\}$ ;
5  if ( $V \neq \emptyset$ ) then
6    Refine :
7      choose  $\langle a, x, x' \rangle \in V$ ;
8       $R(x) := R(x) \cap \text{pre}_a(R(x'))$ ;
9  else
10   return  $\langle R, \sigma \rangle$ ;
11 end if
12 end while

```

**Example IV.5.** Consider the following transition system with infinitely many states where  $I = \{0\}$  is the initial state,  $L$  is a singleton, and the boxes gather states sharing the same principal in  $R_i$ .



We fix the inputs of Algorithm 1 to be  $\sigma_i = \emptyset$  and  $R_i$ , the initial qo is given by  $R_i(0) = \{0\}$ ,  $R_i(1) = \{0, 1\}$ , and  $R_i(n) = ]-\infty, -1]$  for  $n < 0$ . The simulation quasiorder is therefore:  $R_{\text{sim}}(0) = \{0\}$ ,  $R_{\text{sim}}(1) = \{0, 1\}$ , and  $R_{\text{sim}}(n) = ]-\infty, n]$  for each  $n > 0$ . Notice that  $R_{\text{sim}}$  has infinitely many principals. Executing Algorithm 1, we get  $\sigma = \{0, 1\}$  after two *Search* iterations. At this point  $V = \emptyset$  and  $U = \emptyset$ , and the algorithm terminates and returns with the correct result. Now consider instead the process of refining each principal  $R_i(x)$  such that  $R_i(x) \neq R_{\text{sim}}(x)$ . This process, which converges to  $R_{\text{sim}}$ , does not terminate after finitely many steps since infinitely many principals need to be refined. ♦

Let us observe 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. Based on this observation, we define Algorithm 2, also denoted by SymRef, obtained as partial evaluation of Algorithm 1 by assuming to have in input  $\sigma_i = \text{post}^*(I)$ . Moreover, 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, later, we will use Algorithm 2 as a subroutine having in input a relation  $R_i$  that sits in between a quasiorder  $R_{\text{qo}}$  and the simulation preorder  $R_{\text{sim}}$  induced by  $R_{\text{qo}}$ . More precisely, it turns out

**Algorithm 3: Complete Algorithm**

**Input:** A trans. system  $G = (\Sigma, I, L, \rightarrow)$ , an initial qo  $R_i \in \text{QO}(\Sigma)$ , an initial set  $I \subseteq \sigma_i \subseteq \text{post}^*(I)$ .

```

1 Rel( $\Sigma$ )  $\ni R := R_i$ ;
2  $\wp(\Sigma) \ni \sigma := \sigma_i$ ;
3  $\wp(U) \ni U_{\text{bad}} := \emptyset$ ;
4 while true do
  // INV1:  $\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x)$ 
  // INV2:  $\sigma_i \subseteq \sigma \subseteq \text{post}^*(I)$ 
  // INV3:  $\forall x \in \Sigma. x \in R(x)$ 
  // INV4:  $U_{\text{bad}} \subseteq U$ 
5   $U := \{x \in \Sigma \mid \nexists s \in \sigma. R(x) = R(s), R(x) \cap \text{post}(\sigma) \neq \emptyset\}$ ;
6   $V := \{(a, x, x') \in L \times \Sigma^2 \mid \exists s \in \sigma. R(x) = R(s), x \xrightarrow{a} x', R(x) \not\subseteq \text{pre}_a(R(x'))\}$ ;
7  nif
8    ( $U \setminus U_{\text{bad}} \neq \emptyset$ )  $\longrightarrow$  Search :
9    choose  $x \in U \setminus U_{\text{bad}}$ ;
10    $S := (R(x) \cap \text{post}(\sigma)) \setminus \sigma$ ;
11   if  $S \neq \emptyset$  then
12     choose  $s \in S$ ;
13      $\sigma := \sigma \cup \{s\}$ ;
14      $U_{\text{bad}} := \emptyset$ ;
15   else
16      $U_{\text{bad}} := U_{\text{bad}} \cup \{x\}$ ;
17   end if
18   ( $V \neq \emptyset$ )  $\longrightarrow$  Refine :
19   choose  $\langle a, x, x' \rangle \in V$ ;
20    $S := \text{pre}_a(R(x'))$ ;
21    $R(x) := R(x) \cap S$ ;
22    $U_{\text{bad}} := \emptyset$ ;
23   ( $U \setminus U_{\text{bad}} \neq \emptyset \wedge V = \emptyset$ )  $\longrightarrow$  Expand :
24   if  $\text{post}(\sigma) \subseteq \sigma$  then
25     // Run Algo. 2 with input  $R$  and  $\sigma$ 
26     return SymRef( $R, \sigma$ );
27   else
28      $\sigma := \sigma \cup \text{post}(\sigma)$ ;
29      $U_{\text{bad}} := \emptyset$ ;
30   end if
31   ( $U \setminus U_{\text{bad}} \neq \emptyset \wedge V = \emptyset$ )  $\longrightarrow$  return  $\langle R, \sigma \rangle$ ;
32 end while

```

that for Algorithm 2 the invariant  $\text{INV}_1 \triangleq R_{\text{sim}} \subseteq R_i \subseteq R_{\text{qo}}$  holds. Note that  $R_i$  is reflexive since  $R_{\text{sim}}$  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 IV.1 and IV.4.

## V. A COMPLETE REACHABLE SIMULATION ALGORITHM

Algorithm 1 is sound and complete for  $R_{\text{sim}}$  (cf. Theorem IV.1 (1.a)) and, as shown in Example IV.3, in general sound but not complete for  $P_{\text{sim}}$  (cf. Theorem IV.1 (1.b)). To achieve completeness, we perform several key changes

to Algorithm 1 leading to the pseudocode in logical form presented as Algorithm 3. First, we modify the definitions of the sets  $U$  and  $V$  at lines 5 and 6. Informally speaking, the changes made to the definitions of  $U$  and  $V$  reflect the difference between definitions (2) and (3). Moreover, this algorithm keeps track of a subset of  $U$ , denoted by  $U_{bad}$ , that contains states on which executing a *Search* iteration can never expand  $\sigma$ . Furthermore, we added an *Expand* guarded procedure in the loop of nondeterministic choices whose role is to guarantee progress by adding new elements to  $\sigma$ . The *Expand* procedure at line 25 has a return statement invoking Algorithm 2 as a subroutine whenever  $\sigma$  turns out to be the whole set  $\text{post}^*(I)$  of reachable states. It is worth recalling that the results in Section III-B entail that computing the whole set  $\text{post}^*(I)$  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  $\sigma$  are in a 1-1 correspondence with the reachable blocks of  $P_{\text{sim}}$ , and they split every reachable block consistently with  $P_{\text{sim}}$  (cf. (3.b) below). Note that a block  $B \in P^\sigma$  might be strictly contained in the corresponding block  $P_{\text{sim}}(B)$  of  $P_{\text{sim}}$ , but this algorithm ensures that  $P_{\text{sim}}(B) \setminus B \cap \text{post}^*(I) = \emptyset$  (cf. (3.c) below). On the other hand, Algorithm 3 is also complete for the reachable principals of the simulation qo  $R_{\text{sim}}$  where reachability for principals is defined according to definition (3) (cf. (3.a) below).

**Theorem V.1 (Correctness of Algorithm 3).** *Let  $\langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \wp(\Sigma)$  be the output of Algorithm 3 on input  $G$  with  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$ ,  $I \subseteq \sigma_i \subseteq \text{post}^*(I)$ . Let  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation qo and partition w.r.t.  $R_i$ . Let  $P \triangleq \{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma} \in \text{Part}(\Sigma)$ . Then:*

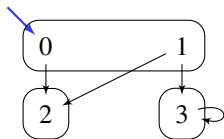
$$\{R_{\text{sim}}(x) \mid x \in \text{post}^*(I)\} = \{R(x) \mid x \in \sigma\}, \quad (3.a)$$

$$\{B \cap \text{post}^*(I) \mid B \in P_{\text{sim}}^{\text{post}^*(I)}\} = \{B \cap \text{post}^*(I) \mid B \in P^\sigma\}, \quad (3.b)$$

$$P_{\text{sim}}^{\text{post}^*(I)} = \{P_{\text{sim}}(B) \mid B \in P^\sigma\}. \quad (3.c)$$

The following example shows the role of the *Expand* guarded procedure and of the call of Algorithm 2 at line 25 in ensuring termination.

**Example V.2.** Consider the following finite transition system, where  $I = \{0\}$ ,  $L$  is a singleton,  $\sigma_i = I$ , and the initial qo  $R_i$  is given by  $R_i(0) = R_i(1) = \{0, 1\}$ ,  $R_i(2) = \{2\}$  and  $R_i(3) = \{2, 3\}$ , where, as before, the boxes gather states sharing the same principal in  $R_i$ .



The simulation quasiorder  $R_{\text{sim}}$  is such that  $R_{\text{sim}}(0) = \{0, 1\}$ ,  $R_{\text{sim}}(1) = \{1\}$ ,  $R_{\text{sim}}(2) = \{2\}$  and  $R_{\text{sim}}(3) = \{3\}$ , meaning

that states 0 and 1 are split in  $P_{\text{sim}}$ . Executing Algorithm 3, we find that  $\sigma = \{0, 2\}$  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 iteration of *Search*. Executing an *Expand* procedure is then unable to enlarge  $\sigma$  since  $\text{post}(\sigma) \subseteq \sigma$ , so that the algorithm will execute the call  $\text{SymRef}(R, \sigma)$  of Algorithm 2 at line 25, thus computing that  $R(1) = \{1\}$ , and in turn separating the states 0 and 1 in  $P$ . ♦

As aforementioned, Algorithm 3 terminates on finite state systems.

**Theorem V.3 (Termination of Algorithm 3).** *Let  $G = (\Sigma, I, L, \rightarrow)$  with  $|\Sigma| \in \mathbb{N}$ ,  $I \subseteq \sigma_i \subseteq \text{post}^*(I)$  and  $R_i \in \text{QO}(\Sigma)$ . Then, Algorithm 3 terminates on input  $G$ ,  $R_i$  and  $\sigma_i$ .*

## VI. 2-PARTITION-RELATION TRIPLES

We define 2-partition-relation triples (2PRs), generalizing the notion of partition-relation pair used by the simulation algorithm in [25] as a symbolic representation of a state relation, and later utilized by the state-of-the-art simulation algorithm in [18]. In Section VII, 2PRs are exploited to design a symbolic version of Algorithm 1. Similar symbolic approaches for simulation algorithms based on state partition are known to be beneficial for algorithms manipulating infinite state systems, as shown by Henzinger et al.'s symbolic simulation algorithm for infinite graphs and, in particular, hybrid automata [16], and have been proven to be advantageous in terms of space and time efficiency also for finite state systems [18], [19]. The rationale behind the need for 2PRs as a generalization has more to do with improving the presentation and ease of understanding and less to do with fundamental limitations partition-relation pairs.

**Definition VI.1 (2PR Triple).** Given a (possibly infinite) set  $\Sigma$ , a triple  $\langle P, \tau, Q \rangle$ , where  $P, Q \in \text{Part}(\Sigma)$  and  $\tau: P \rightarrow \wp(Q)$ , is a 2-partition-relation (2PR) triple. ♦

A relation  $R \in \text{Rel}(\Sigma)$  induces a 2PR triple  $\langle P_R, \tau_R, Q_R \rangle$  as follows:

$$\begin{aligned} P_R &\triangleq \{y \in \Sigma \mid R(x) = R(y)\}_{x \in \Sigma}, \\ Q_R &\triangleq \{y \in \Sigma \mid R^{-1}(x) = R^{-1}(y)\}_{x \in \Sigma}, \\ \tau_R(B) &\triangleq \{C \in Q_R \mid C \subseteq R(B)\}. \end{aligned}$$

The idea of maintaining two separate partitions  $P$  and  $Q$  is for  $P$  to keep track of states having the same principal, and for  $Q$  to keep track of states occurring in the same set of principals.

Conversely, a 2PR triple  $\langle P, \tau, Q \rangle$  encodes a relation  $R_{\langle P, \tau, Q \rangle} \in \text{Rel}(\Sigma)$  defined as  $R_{\langle P, \tau, Q \rangle}(x) \triangleq \bigcup \tau(P(x))$  and a partition  $P_{\langle P, \tau, Q \rangle} \triangleq \{y \in \Sigma \mid \bigcup \tau(P(y)) = \bigcup \tau(P(x))\}_{x \in \Sigma} \in \text{Part}(\Sigma)$ , such that  $P_{\langle P, \tau, Q \rangle}$  is a coarser partition than  $P$ .

**Property VI.2.** *If  $R \in \text{Rel}(\Sigma)$  (no assumption on  $R$ ), then  $R = R_{\langle P_R, \tau_R, Q_R \rangle}$  holds.*

Moreover, 2PR triples encoding reflexive relations can be characterized by extensivity of  $\lambda B \in P. \bigcup \tau(B)$ :

---

**Algorithm 4: Sound Symbolic Algorithm**

---

**Input:** A transition system  $G = (\Sigma, I, L, \rightarrow)$ , an initial  
    go  $R_i \in \text{QO}(\Sigma)$ , an initial set  $\sigma_i \subseteq \text{post}^*(I)$

```
1 Part( $\Sigma$ )  $\ni P := \{y \in \Sigma \mid R_i(x) = R_i(y)\}_{x \in \Sigma};$   
2 Part( $\Sigma$ )  $\ni Q := \{y \in \Sigma \mid R_i^{-1}(x) = R_i^{-1}(y)\}_{x \in \Sigma};$   
3 forall  $B \in P$  do  
     $\wp(Q) \ni \tau(B) := \{C \in Q \mid C \subseteq R_i(B)\};$   
4  $\wp(\Sigma) \ni \sigma := \sigma_i;$   
5 while true do  
    // INV1:  $\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R_{\langle P, \tau, Q \rangle}(x) \subseteq R_i(x)$   
    // INV2:  $\sigma_i \subseteq \sigma \subseteq \text{post}^*(I)$   
    // INV3:  $\forall B \in P. B \subseteq \cup \tau(B)$   
6  $U := \{B \in P \mid \cup \tau(B) \cap \sigma = \emptyset,$   
     $\cup \tau(B) \cap (I \cup \text{post}(\sigma)) \neq \emptyset\};$   
7  $V := \{\langle a, B, C \rangle \in L \times P^2 \mid \cup \tau(B) \cap \sigma \neq \emptyset,$   
     $B \cap \text{pre}_a(C) \neq \emptyset, \cup \tau(B) \not\subseteq \text{pre}_a(\cup \tau(C))\};$   
8 nif  
9  $(U \neq \emptyset) \rightarrow \text{Search} :$   
10 choose  $B \in U;$   
11 choose  $s \in (\cup \tau(B) \cap (I \cup \text{post}(\sigma)));$   
12  $\sigma := \sigma \cup \{s\};$   
13  $(V \neq \emptyset) \rightarrow \text{Refine} :$   
14 choose  $\langle a, B, C \rangle \in V; S := \text{pre}_a(\cup \tau(C));$   
15  $B' := B \cap \text{pre}_a(C); B'' := B \setminus \text{pre}_a(C);$   
16  $P.\text{replace}(B, \{B', B''\});$   
17  $\tau(B') := \tau(B); \tau(B'') := \tau(B);$   
18 forall  
     $X \in \{E \in \tau(B') \mid E \cap S \neq \emptyset, E \not\subseteq S\}$  do  
         $Q.\text{replace}(X, \{X \cap S, X \setminus S\});$   
19 foreach  $A \in P$  do  
     $\tau(A).\text{replace}(X, \{X \cap S, X \setminus S\});$   
21 end forall  
22  $\tau(B') := \{E \in \tau(B') \mid E \subseteq S\};$   
23  $(U = \emptyset \wedge V = \emptyset) \rightarrow \text{return } \langle P, \tau, Q, \sigma \rangle;$   
24 fin  
25 end while
```

---

**Property VI.3.** Let  $\langle P, \tau, Q \rangle$  be a 2PR triple over  $\Sigma$ . Then:

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

Finally, it turns out that 2PR triples induced by quasiorders enjoy the following property:

**Property VI.4.** Let  $R$  be a go and  $\langle P_R, \tau_R, Q_R \rangle$  be the induced 2PR triple. Then,  $P_R = Q_R$ .

## VII. 2PR ALGORITHM

We now put forward Algorithm 4, designed as a revision of Algorithm 1 based on representing the relation  $R$  as a 2PR triple. Algorithm 4 is symbolic in the sense that it symbolically represents and processes state relations as 2PR triples  $\langle P, \tau, Q \rangle$ . Since the refinement process preserves reflexivity of the underlying relation (as it happens for Algorithm 1), we have that Property VI.3 ensures that  $\lambda B \in P. \cup \tau(B)$  is

extensive. During its execution, for each state  $x$ ,  $R_{\langle P, \tau, Q \rangle}(x)$  is a set of states candidates to simulate  $x$ , and all the states in the same block of  $P_{\langle P, \tau, Q \rangle}(x)$  are candidates to be simulation equivalent.

Following the approach of Algorithm 1, this symbolic algorithm computes a set of principals  $\{\cup \tau(B) \mid \cup \tau(B) \cap \sigma \neq \emptyset\}_{B \in P}$  which are known to contain a reachable state. A block  $B$  is in  $U$  if  $\cup \tau(B)$  does not contain states that are currently known to be reachable while it contains either an initial state or a successor of a state which is known to be reachable. Also, the set  $V$  contains unstable symbolic triples:  $\langle a, B, C \rangle$  is in  $V$  iff  $\cup \tau(B)$  is known to contain some reachable state and there exists  $b \in B, c \in C$  such that  $R_{\langle P, \tau, Q \rangle}(b)$  is  $a$ -unstable w.r.t.  $R_{\langle P, \tau, Q \rangle}(c)$ . The algorithm either updates the reachability information for some principal of a block in  $U$  executing the *Search* procedure or stabilizes the pair of blocks associated to some triple in  $V$  executing the *Refine* procedure. The *Refine* procedure  $a$ -stabilizes a pair of blocks  $(B, C)$  by possibly splitting  $B$  into  $B \cap \text{pre}_a(C)$  and  $B \setminus \text{pre}_a(C)$  (lines 15–17), and then refines the principal  $\cup \tau(B \cap \text{pre}_a(C))$  at lines 18–21 by first splitting blocks of  $Q$  if they are known to contain states occurring in different sets of principals (for the current underlying relation  $R_{\langle P, \tau, Q \rangle}$ ), and successively by removing at line 22 all the blocks not contained in  $\text{pre}_a(\cup \tau(C))$ . Upon termination,  $R_{\langle P, \tau, Q \rangle}^\sigma$  coincides with the set of reachable principals of  $R_{\text{sim}}$  (cf. (3.a) below). Turning to simulation partition, it turns out that Algorithm 4 yields a sound overapproximation of  $P_{\text{sim}}^{\text{post}^*(I)}$  (cf. (3.b) below). The proof of correctness for Algorithm 4 is meaningful and we include it in the main body of the paper, along with the auxiliary basic simulation Algorithm 5 which is used in the proof.

---

**Algorithm 5: Sim**

---

**Input:** A transition system  $G = (\Sigma, I, L, \rightarrow)$ , a  
    relation  $R_i \in \text{Rel}(\Sigma)$  s.t.  $R_{\text{sim}} \subseteq R_i \subseteq R_{qo}$  for  
    some go  $R_{qo}$ , and the simulation go  $R_{\text{sim}}$   
    induced by  $R_{qo}$ .

```
1 Rel( $\Sigma$ )  $\ni R := R_i;$   
2 while  $\exists x, x' \in \Sigma, a \in L. x \xrightarrow{a} x' \wedge R(x) \not\subseteq \text{pre}_a(R(x'))$   
    do  
        // INV1:  $\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x)$   
3  $R(x) := R(x) \cap \text{pre}_a(R(x'));$   
4 end while  
5 return  $R;$ 
```

---

**Theorem VII.1 (Correctness of Algorithm 4).** Let  $\langle P, \tau, Q, \sigma \rangle$  be the output of Algorithm 4 on input  $G$  with  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$  and  $\sigma_i \subseteq \text{post}^*(I)$ . Let  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation go and partition



w.r.t.  $R_i$ . Then:

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

$$P_{\text{sim}}^{\text{post}^*(I)} \subseteq \{B \in P_{\langle P, \tau, Q \rangle} \mid \exists E \in P. E \subseteq B \wedge (\cup \tau(E)) \cap \sigma \neq \emptyset\}. \quad (3.b)$$

*Proof:* 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 quasiorder and we proceed to 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  $\forall X \in P. X \not\subseteq B \Rightarrow \cup \tau(X) = \cup \tau'(X)$ , and that  $\cup \tau(P(B'')) = \cup \tau'(B'')$  since the only instruction removing states from  $\cup \tau(X)$ , for some block  $X$  is line 22. We now take  $x \in B' = B \cap \text{pre}_a(C)$  and proceed to show that  $x \in \cup \tau'(B')$ . Since  $x \in \text{pre}_a(C) \subseteq \text{pre}_a(\cup \tau(C))$  then  $x \in S$  and thus  $Q(x) \cap S \neq \emptyset$ . We now distinguish two cases in order 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, then we get  $Q'(x) \in \tau(B')$  after line 22 and thus 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  $R_{\langle P_{R_i}, \tau_{R_i}, Q_{R_i} \rangle} = R_i$ , and we proceed to 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  $y \in \cup \tau(P(x)) \setminus \cup \tau'(P'(x))$  it holds that  $y \notin R_{\text{sim}}(x)$ .

In order to do so we observe that  $Q'(y) \notin \tau'(P'(x))$  implies that line 22 removed the block  $Q'(y)$ , thus meaning  $Q'(y) \not\subseteq S$ . 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  $Q'(y) \subseteq Q(y)$  then  $Q(y) \not\subseteq S$ , meaning the block  $Q(y)$  was split during the *Refine* step, meaning that either  $Q'(y) = Q(y) \cap S$  or  $Q'(y) = Q(y) \setminus S$ . We note that, the case  $Q'(y) = Q(y) \cap S$  leads to a contradiction since  $Q'(y) \not\subseteq S$  and thus we

infer  $Q'(y) = Q(y) \setminus S$ . Observing that  $y \in Q'(y)$  lets us conclude  $y \notin S$ .

We now observe that, since  $x \in B \cap \text{pre}_a(C)$ , then there exists some  $x' \in C$  s.t.  $x \xrightarrow{a} x'$ . 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  $x \rightarrow x'$  implies  $y \notin R_{\text{sim}}(x)$ .

- (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  $I \xrightarrow{n} x$ . If  $n = 0$  then  $x \in I$ , so that, since  $P(x) \subseteq \cup \tau(P(x))$  following (i), and  $x \in P(x)$ , then  $\cup \tau(P(x)) \cap (I \cup \text{post}(\sigma)) \neq \emptyset$ . Hence,  $U = \emptyset$  implies that  $\cup \tau(P(x)) \cap \sigma \neq \emptyset$ . If  $n > 0$  then  $I \xrightarrow{n} x' \xrightarrow{a} x$ , 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  $\text{Sim}$  be the basic simulation algorithm as recalled in Algorithm 5. 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  $\text{Sim}$  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  $\text{Sim}$  before this iteration  $it$ . Let  $R'(y) \subseteq R_{\langle P, \tau, Q \rangle}(y)$  be the principal of  $R'$  used by  $\text{Sim}$  in this iteration  $it$  to refine  $R_{\langle P, \tau, Q \rangle}(x)$ , so that  $x \xrightarrow{a} y$ ,  $R'(x) = R_{\langle P, \tau, Q \rangle}(x) \not\subseteq \text{pre}_a(R'(y))$ . By termination of Algorithm 4 it holds that  $R_{\langle P, \tau, Q \rangle}(x) \cap \sigma \neq \emptyset$ ,  $P(x) \cap \text{pre}(P(y)) \neq \emptyset$  and  $V = \emptyset$  entail  $\cup \tau(P(x)) \subseteq \text{pre}_a(\cup \tau(P(y)))$ . 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  $\text{Sim}$ , 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  $R_{\langle P, \tau, Q \rangle}(x) \not\subseteq \text{pre}_a(R_{\langle P, \tau, Q \rangle}(y))$ , which gives a contradiction to  $\cup \tau(P(x)) \subseteq \text{pre}_a(\cup \tau(P(y)))$ . Thus,  $\mathbb{S} = \emptyset$ , and, in turn,  $R_{\langle P, \tau, Q \rangle}(x) = R_{\text{sim}}(x)$ .

- (v) At each iteration:  $\sigma \subseteq \text{post}^*(I)$ .

This follows by  $\sigma_i \subseteq \text{post}^*(I)$  and because  $\sigma$  is updated at line 12 of Algorithm 4 by adding  $s \in I \cup \text{post}(\sigma)$  and  $\text{post}^*(I)$  is the least fixpoint of  $\lambda X. I \cup \text{post}(X)$ .

- (vi) At termination:  $P_{\langle P, \tau, Q \rangle}(x) \cap \sigma \neq \emptyset \Rightarrow P_{\langle P, \tau, Q \rangle}(x) = P_{\text{sim}}(x)$ .

We first observe that, assuming  $P_{\langle P, \tau, Q \rangle}(x) \cap \sigma \neq \emptyset$ , 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  $R_{\langle P, \tau, Q \rangle}(s) \cap \sigma \neq \emptyset$ . Therefore, by (iv),  $R_{\text{sim}}(x) = R_{\langle P, \tau, Q \rangle}(x) = R_{\langle P, \tau, Q \rangle}(s) = R_{\text{sim}}(s)$ . We prove the two inclusions of  $P_{\langle P, \tau, Q \rangle}(x) = P_{\text{sim}}(x)$  separately:

( $\subseteq$ ): Assume  $y \in P_{\langle P, \tau, Q \rangle}(x)$ , then  $\cup \tau(P(y)) = \cup \tau(P(x))$  and since  $\cup \tau(P(x)) \cap \sigma \neq \emptyset$  then  $\cup \tau(P(y)) \cap \sigma \neq \emptyset$ . Finally, (iv) entails  $R_{\text{sim}}(y) = \cup \tau(P(y)) = \cup \tau(P(x)) = R_{\text{sim}}(x)$  thus proving  $y \in P_{\text{sim}}(x)$ .

( $\supseteq$ ): Assume  $y \in P_{\text{sim}}(x)$ , then  $R_{\text{sim}}(y) = R_{\text{sim}}(x) = R_{\text{sim}}(s)$ . Moreover, since  $s \in \sigma$ , and  $s \in R_{\text{sim}}(s)$ , we get  $R_{\text{sim}}(y) \cap \sigma \neq \emptyset$  and since, by (ii),  $R_{\text{sim}}(y) \subseteq R_{\langle P, \tau, Q \rangle}(y)$  then  $R_{\langle P, \tau, Q \rangle}(y) \cap \sigma \neq \emptyset$ . Therefore, (iv) entails  $R_{\langle P, \tau, Q \rangle}(y) = R_{\text{sim}}(y)$ , and thus  $R_{\langle P, \tau, Q \rangle}(y) = R_{\text{sim}}(y) = R_{\text{sim}}(x) = R_{\langle P, \tau, Q \rangle}(x)$  holds, proving  $\cup \tau(P(y)) = \cup \tau(P(x))$  and thus  $y \in P_{\langle P, \tau, Q \rangle}(x)$ .

Let us now show the equality (3.a).

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

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

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

Therefore,  $P_{\text{sim}}(x) \subseteq \{y \in \Sigma \mid \cup \tau(P(x)) = \cup \tau(P(y))\} = P_{\langle P, \tau, Q \rangle}(x)$ . 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  $\cup \tau(P(x)) \cap \sigma \neq \emptyset$  implies  $R_{\langle P, \tau, Q \rangle}(z) \cap \sigma \neq \emptyset$ . Thus, by (3.a),  $R_{\text{sim}}(z) = R_{\langle P, \tau, Q \rangle}(z)$ . Hence,  $R_{\text{sim}}(z) = R_{\langle P, \tau, Q \rangle}(z) = R_{\langle P, \tau, Q \rangle}(x) = R_{\text{sim}}(x)$ , thus proving that  $P_{\langle P, \tau, Q \rangle}(x) \subseteq P_{\text{sim}}(x)$ , and therefore  $P_{\langle P, \tau, Q \rangle}(x) = P_{\text{sim}}(x)$  holds. Finally, since  $R_{\langle P, \tau, Q \rangle}(x) \cap \sigma \neq \emptyset$ , then  $\cup \tau(P(x)) \cap \sigma \neq \emptyset$ . Therefore, since  $P_{\langle P, \tau, Q \rangle}$  is coarser than  $P$ , then it holds  $P(x) \subseteq P_{\langle P, \tau, Q \rangle}(x)$ , thus proving  $\exists E \in P. E \subseteq P_{\langle P, \tau, Q \rangle}(x) \wedge (\cup \tau(E)) \cap \sigma \neq \emptyset$  and consequently (1.b) holds. ■

We now show how the following assumption over Algorithm 5 can be leveraged to extend the correctness proofs for our algorithms to infinite transition systems:

**Assumption VII.2.** *If  $y \in R_i(x) \setminus R_{\text{sim}}(x)$  for some  $x \in \Sigma$ , then there exists an execution (not necessarily terminating) of*

*Sim over input  $R_i$  such that, after a finite number of iterations,  $y \notin R(x)$  holds.*

**Remark VII.3 (On the Finiteness Assumption).** Theorems VII.1, V.1 and IV.1 all share a finiteness assumption for the input transition system, which can be lifted under a reasonable assumption. All three proofs share a common pattern and the hypothesis on finiteness of the input system is leveraged in order to conclude that executing Algorithm 5 on such a transition system will converge after finitely many steps. The proofs are thus based on termination of Algorithm 5 and on its correctness (that is, it outputs  $R_{\text{sim}}$ ). It turns out that termination can be replaced by a fairness requirement over all possible executions of Algorithm 5, as stated in Assumption VII.2. More in detail, we require that for each pair  $(x, y) \in R_i$  of states such that  $y$  does not simulate  $x$ , there exists an execution of Algorithm 5 which removes the pair  $(x, y)$  from the current relation  $R$  after finitely many refinement steps. Moreover, by replacing the termination requirement into a fairness one, we replace the correctness assumption on the output (i.e. that Algorithm 5 eventually outputs  $R_{\text{sim}}$ ) by  $\text{Inv}_1$ , which is sufficient for our proof. More details for this extension, (especially pointers to the full proofs for the three theorems) can be found in Appendix C. ♦

**Theorem VII.4 (Termination of Algorithm 4).** *Let  $G = (\Sigma, I, L, \rightarrow)$  with  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$  and  $\sigma_i \subseteq \text{post}^*(I)$ . Then, Algorithm 4 terminates on input  $G$ ,  $R_i$ , and  $\sigma_i$ .*

*Proof:* We first observe that each *Search* iteration adds some new state to  $\sigma$  through the update at line 12. Thus, since  $\Sigma$  has finitely many elements, Algorithm 4 will always execute a finite number of *Search* iterations. We now proceed to show that the algorithm executes a finite number of *Refine* iterations too. Consider a *Refine* iteration of the algorithm, the corresponding triple  $\langle a, B, C \rangle$  selected for refinement, the 2PR triple  $\langle P, \tau, Q \rangle$  before executing the refinement block and the 2PR triple  $\langle P', \tau', Q' \rangle$  obtained right after the refinement step has been executed, then

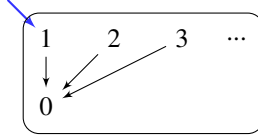
$$\forall x \in B \cap \text{pre}_a(C) \Rightarrow R_{\langle P', \tau', Q' \rangle}(x) \subset R_{\langle P, \tau, Q \rangle}(x) . \quad (4)$$

In fact, let  $x \in B \cap \text{pre}_a(C) = B'$  and  $x' \in C$  be s.t.  $x \xrightarrow{a} x'$ , then by definition of  $V$  we get  $\cup \tau(P(x)) \not\subseteq \text{pre}_a(\cup \tau(P(x')))$ . Assume  $y \in \cup \tau(P(x)) \setminus \text{pre}_a(\cup \tau(P(x')))$  and observe that either  $Q(y) \cap \text{pre}_a(\cup \tau(C)) = \emptyset$ , therefore  $Q(y) = Q'(y)$  will not be split and it will be removed at line 22, or  $Q(y) \cap \text{pre}_a(\cup \tau(C)) \neq \emptyset$ , therefore  $Q(y)$  will be split between lines 18–21 and by definition  $Q'(y) = Q(y) \setminus \text{pre}_a(\cup \tau(C))$  is such that  $Q'(y)$  will be removed at line 22. Therefore,  $Q'(y) \not\subseteq \text{pre}_a(\cup \tau(C)) = S$ , thus showing that the execution of line 22 effectively refines  $\tau(B)$  by removing (some sub-block of)  $Q(y)$  from it, proving (4). Therefore, executing the *Refine* block is guaranteed to remove some state from at least one principal of the underlying relation  $R_{\langle P, \tau, Q \rangle}$ , and since each principal has an initial finite number of 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* step, the algorithm executes a finite number of iterations. Finally, we observe that each iteration computes a finite number of operations, so that this completes the proof. ■

As it is the case for Algorithms 1 and 3, termination holds for finite state systems, and Algorithm 4 may terminate on infinite state systems as well, as shown below in Example VII.6. Moreover, the use of 2PR triples as a representation structure brings benefits in terms of termination on infinite systems, since there exist inputs on which Algorithm 1 does not terminate but Algorithm 4 does. For instance, Algorithm 1 terminates for no input where  $R_i$  is such that the set of principals given by  $\{R_i(x) \mid x \in \Sigma, R_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset, R_{\text{sim}}(x) \neq R_i(x)\}$  is infinite. On such inputs Algorithm 1 executes the *Refine* block infinitely often, as shown in Example VII.5.

**Example VII.5.** Consider the following infinite transition system, where  $\Sigma = \mathbb{N}$ ,  $I = \{1\}$ ,  $L$  is a singleton,  $\sigma_i = I$ , and the initial qo  $R_i$  is  $\mathbb{N} \times \mathbb{N}$ .



The simulation quasiorder  $R_{\text{sim}}$  is such that  $R_{\text{sim}}(0) = \mathbb{N}$  and  $R_{\text{sim}}(n) = \mathbb{N} \setminus \{0\}$  for  $n \geq 1$ , meaning that  $P_{\text{sim}} = \{\{0\}, \{1, 2, \dots\}\}$ . 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 positive integer  $i > 0$ . Each time a *Refine* iteration is executed, exactly one principal is updated. Assume that a triple  $\langle a, i, 0 \rangle \in V$  is selected, so that the refinement  $R(i) = \mathbb{N} \setminus \{0\}$  will be performed. As a consequence, the execution will never terminate because  $V$  initially contains an infinitely many triples, 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 step of *Refine*. To illustrate that, we consider the input from Example VII.5 and show that Algorithm 4 converges in a few iterations.

**Example VII.6.** Consider the same input as in Example VII.5. Here, Algorithm 4 terminates. The initial 2PR triple is given by  $P = Q = \{\mathbb{N}\}$  and  $\tau(\mathbb{N}) = \{\mathbb{N}\}$ , the set  $U$  is empty, and the set  $V$  is  $\{\langle a, \mathbb{N}, \mathbb{N} \rangle\}$ . Executing one iteration refines both  $P$ ,  $Q$  and  $\tau$ , so that  $P = Q = \{\{0\}, \mathbb{N} \setminus \{0\}\}$ ,  $\tau(\{0\}) = P$  and  $\tau(\mathbb{N} \setminus \{0\}) = \{\mathbb{N} \setminus \{0\}\}$ . At this point, the underlying relation is such that  $R_{\langle P, \tau, Q \rangle} = R_{\text{sim}}$ , so that the execution of Algorithm 4 terminates. ♦

Finally, we remark that Algorithm 4 initializes the partition  $P$  and  $Q$  of the 2PR triple such 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 block is a singleton. Algorithm 4 with that initialization remains correct and is not much different from Algorithm 1 since the relation  $R$  of Algo. 1 and  $\tau$  of Algo. 4 are essentially the same.

Coming back to the goal of having  $P$  and  $Q$  as coarse as possible we next show that, when Algorithm 4 executes, each refinement to  $P$  (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  $R_{\langle P, \tau, Q \rangle}$ ). Formally this is stated as follows.

**Property VII.7.** Let  $\langle P, \tau, Q, \sigma \rangle$  be the output of Algorithm 4 with input  $G$ ,  $R_i \in \text{QO}(\Sigma)$  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 let  $\langle P_j, \tau_j, Q_j \rangle$  be the 2PR triple after  $j = 0, \dots, t$  iterations of the algorithm, and  $R_j = R_{\langle P_j, \tau_j, Q_j \rangle}$  be the corresponding underlying relation. Then:

$$P(x) \neq P(y) \Rightarrow R_j(x) \neq R_j(y) \text{ for some } j \in [0, t] , \quad (5)$$

and a dual property for  $Q$  holds:

$$Q(x) \neq Q(y) \Rightarrow R_j^{-1}(x) \neq R_j^{-1}(y) \text{ for some } j \in [0, t] . \quad (6)$$

Next we show that, besides contrived examples like Example VII.6, experimental evidence suggests that Algorithm 4 terminates faster in practice.

## VIII. EXPERIMENTAL EVIDENCE FOR 2PR TRIPLES

We implemented the algorithms presented in this paper for finite transition systems that are explicitly represented. In this section, by leveraging this implementation, we empirically compare Algorithm 1, which uses an explicit representation for  $R$ , and Algorithm 4, which symbolically represents  $R$  through 2PR triples. For the benchmarks we used a small subset of the BEEM database for explicit model checkers [30]. The selected subset of benchmarks consists of mutual exclusion protocols that we modified to increase the number of unreachable states by performing an “unrolling” of the protocol loop and then by picking an initial state after the unrolled protocols. The rationale behind this modification is that, typically, benchmarks available online as explicit transition systems have no unreachable states and, often, are strongly connected, which means that moving the initial state has no effect on the number of unreachable states.

Our experimental results are summarized in Table I where the initial equivalence/quasiorder is set to  $R_i = \Sigma \times \Sigma$ , and  $\sigma_i = \emptyset$ .

Following their definitions, we infer that the following inequalities must hold for the entries of Table I:

$$\Sigma \geq P^1 \geq r^1 , \text{ and } \Sigma \geq P^4 \geq P_{P, \tau, Q}^4 \geq r^4 .$$

The results in Table I show that Algorithm 4 has a significant gain in term of execution time (reduction ranges between



TABLE I

COMPARISON OF ALGORITHMS 1 AND 4. THE COLUMN  $\rightarrow$  IS THE NUMBER OF LABELED TRANSITIONS;  $\Sigma$  THE SIZE OF THE STATE SPACE;  $\sigma^1$  THE SIZE OF  $\sigma$  AS COMPUTED BY ALGORITHM 1,  $P^1$  THE SIZE OF  $P$  AS COMPUTED BY ALGORITHM 1 AS DEFINED IN THEOREM IV.1, AND  $t^1$  IS THE EXECUTION TIME IN SECONDS;  $r^1$  IS THE NUMBER OF BLOCKS IN THE OUTPUT OF ALGORITHM 1 AS PER THE RIGHT HAND SIDE OF (1.b):  $|\{B \in P \mid R(B) \cap \sigma \neq \emptyset\}|$ ;  $\sigma^4$ ,  $P^4$ ,  $Q^4$ ,  $P^4_{(P,\tau,Q)}$  THE SIZES OF  $\sigma$ ,  $P$ ,  $Q$  AND  $P_{(P,\tau,Q)}$ , RESPECTIVELY, AS COMPUTED BY ALGORITHM 4, AND  $t^4$  IS THE EXECUTION TIME IN SECONDS;  $r^4$  IS THE NUMBER OF BLOCKS IN THE OUTPUT OF ALGORITHM 4 AS PER THE RIGHT HAND SIDE OF (3.b):  $|\{B \in P_{(P,\tau,Q)} \mid \exists E \in P. E \subseteq B \wedge (\cup \tau(E)) \cap \sigma \neq \emptyset\}|$ ; **gain** IS THE PERCENTAGE DECREASE IN EXECUTION TIME OF ALGORITHM 4 W.R.T. ALGORITHM 1, I.E.  $(t^1 - t^4)/t^1$ . THE SYMBOL  $\dagger$  INDICATES EXECUTION TIMEOUT AFTER 24 HOURS.

Protocol	$\rightarrow$	$\Sigma$	$\sigma^1$	$P^1$	$r^1$	$t^1$	$\sigma^4$	$P^4$	$Q^4$	$P^4_{(P,\tau,Q)}$	$r^4$	$t^4$	gain %
bakery.1	2917	1628	60	464	396	246.73	59	672	486	466	396	28.66	88.4
bakery.2	2453	1340	96	769	638	145.28	99	779	752	741	638	72.51	50.1
fischer.1	2748	1254	86	359	295	117.17	91	675	512	511	295	62.21	46.9
fischer.2	131407	42272	$\dagger$	$\dagger$	$\dagger$	$\dagger$	$\dagger$	$\dagger$	$\dagger$	$\dagger$	$\dagger$	$\dagger$	—
mcs.1	31645	11839	165	1185	447	59333.74	172	766	793	661	447	253.93	99.6
mcs.2	480	208	39	39	39	0.72	37	42	39	39	39	0.07	89.9
peterson.1	47064	17919	$\dagger$	$\dagger$	$\dagger$	$\dagger$	928	2012	2176	1787	1280	4211.78	$\infty$

a minimum of 46.9% and a maximum of 99.6% less time required for computation). Of course, it is worth pointing out that the set of benchmarks is fairly small and that every transition system is a mutual exclusion protocol that has been unrolled.

## IX. CONCLUSION AND FUTURE WORK

We introduced and proved the soundness of several algorithms with different trade-offs, assumptions and limitations for the reachable simulation problem, which discloses some fundamental differences in decidability and complexity w.r.t. to the analogous problem for bisimulation studied by Lee and Yannakakis [3]. To the best of our knowledge, this is the first investigation of this problem. Algorithm 4 is the most relevant one for practical purposes since it converges on all finite state systems and is well-suited to handle infinite state systems thanks to its symbolic representation, being able to converge on some—but not all, due to an undecidability result which we have shown—such transition systems.

Future work includes, but is not limited to, the following tasks:

- To investigate the reachable simulation problem for transition systems with infinitely many states, therefore assuming an implicit representation of the transition system as well as effective means to represent and process the state relation  $R$ . Such an example of infinite transitions systems could be programs with variables ranging over an infinite data domain equipped with a dedicated logical formalism for representing  $R$ . Another one could be timed automata and their regions/zone representation of the state space. A possible starting point could be the notion of “region algebra” of a transition system as put forward by Henzinger, Majumdar and Raskin [31]. Note that Appendix C makes a step towards dealing with transition systems with infinitely many states by showing how to extend correctness results for the infinite case under some assumptions.
- To establish upper bounds of the algorithms in terms of number of basic operations performed on a representa-

tion of  $R$ , such as the aforementioned region algebra, following the approach of Lee and Yannakakis [3, §3].

- To study algorithmic improvements of our algorithms, for instance, by leveraging the data structures such as queue and stacks used by Lee and Yannakakis’ algorithm for bisimulation.

## REFERENCES

- [1] D. Bustan and O. Grumberg, “Simulation-based minimization,” *ACM Trans. Comput. Log.*, vol. 4, no. 2, pp. 181–206, 2003.
- [2] R. van Glabbeek and B. Ploeger, “Five determinisation algorithms,” in *Implementation and Applications of Automata*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 161–170.
- [3] D. Lee and M. Yannakakis, “Online Minimization of Transition Systems,” in *Proc. of the 24th Annual ACM Symposium on Theory of Computing, STOC ’92*. ACM, 1992, pp. 264–274.
- [4] A. Bouajjani, J.-C. Fernandez, and N. Halbwachs, “Minimal model generation,” in *Proc. of the International Workshop on Computer-Aided Verification, CAV’91*, ser. LNCS. Springer, 1991, pp. 197–203.
- [5] A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel, “Minimal state graph generation,” *Science of Computer Programming*, vol. 18, no. 3, pp. 247–269, 1992.
- [6] R. Alur and T. A. Henzinger, “Computer-Aided Verification,” 1999, chapter 4: Graph Minimization.
- [7] K. Fisler and M. Y. Vardi, “Bisimulation minimization and symbolic model checking,” *Formal Methods Syst. Des.*, vol. 21, no. 1, pp. 39–78, 2002.
- [8] B. S. Gulavani, T. A. Henzinger, Y. Kannan, A. V. Nori, and S. K. Rajamani, “SYNERGY: a new algorithm for property checking,” in *Proc. of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006*. ACM, 2006, pp. 117–127.
- [9] C. S. Pasareanu, R. Pelánek, and W. Visser, “Concrete model checking with abstract matching and refinement,” in *Proc. 17th International Conference on Computer Aided Verification, CAV 2005*, ser. LNCS. Springer, 2005, pp. 52–66.
- [10] R. Majumdar, N. Ozay, and A.-K. Schmuck, “On abstraction-based controller design with output feedback,” in *Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control, HSCC 2020*. ACM, 2020, pp. 1–11.
- [11] S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis, “Property preserving simulations,” in *Proc. of the Fourth International Workshop on Computer Aided Verification, CAV’92*, ser. LNCS. Springer, 1992, pp. 260–273.
- [12] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*, 1st ed. Springer, 2018.
- [13] O. Grumberg and D. E. Long, “Model checking and modular verification,” in *Proc. of the 2nd International Conference on Concurrency Theory, CONCUR’91*, ser. LNCS. Springer, 1991, pp. 250–265.



- [14] —, “Model checking and modular verification,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 3, pp. 843–871, 1994.
- [15] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem, “Property preserving abstractions for the verification of concurrent systems,” *Formal Methods Syst. Des.*, vol. 6, no. 1, pp. 11–44, 1995.
- [16] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, “Computing simulations on finite and infinite graphs,” in *Proc. of IEEE 36th Annual Foundations of Computer Science, FOCS’95*, 1995, pp. 453–462.
- [17] B. Bloom and R. Paige, “Transformational design and implementation of a new efficient solution to the ready simulation problem,” *Sci. Comput. Program.*, vol. 24, no. 3, pp. 189–220, 1995.
- [18] G. Cécé, “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, 2017, pp. 1–12.
- [19] S. Crafa, F. Ranzato, and F. Tapparo, “Saving space in a time efficient simulation algorithm,” *Fundam. Informaticae*, vol. 108, no. 1-2, pp. 23–42, 2011.
- [20] R. Gentilini, C. Piazza, and A. Policriti, “From bisimulation to simulation: Coarsest partition problems,” *Journal of Automated Reasoning*, vol. 31, no. 1, pp. 73–103, 2003.
- [21] R. v. Glabbeek and B. Ploeger, “Correcting a space-efficient simulation algorithm,” in *Proc. of the International Conference on Computer Aided Verification, CAV’08*. Springer, 2008, pp. 517–529.
- [22] F. Ranzato, “A more efficient simulation algorithm on Kripke structures,” in *Proc. of the 38th International Symposium on Mathematical Foundations of Computer Science 2013, MFCS 2013*, ser. LNCS. Springer, 2013, pp. 753–764.
- [23] —, “An efficient simulation algorithm on Kripke structures,” *Acta informatica*, vol. 51, no. 2, pp. 107–125, 2014.
- [24] F. Ranzato and F. Tapparo, “An efficient simulation algorithm based on abstract interpretation,” *Information and Computation*, vol. 208, no. 1, pp. 1–22, 2010.
- [25] —, “A new efficient simulation equivalence algorithm,” in *Proc. of the 22nd IEEE Symposium on Logic in Computer Science, LICS 2007*. IEEE Computer Society, 2007, pp. 171–180.
- [26] L. Tan and R. Cleaveland, “Simulation revisited,” in *Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001*, ser. LNCS. Springer, 2001, pp. 480–495.
- [27] A. Kucera and R. Mayr, “Simulation preorder over simple process algebras,” *Inf. Comput.*, vol. 173, no. 2, pp. 184–198, 2002.
- [28] —, “Why is simulation harder than bisimulation?” in *Proc. of the 13th International Conference on Concurrency Theory, CONCUR 2002*, ser. LNCS. Springer, 2002, pp. 594–610.
- [29] M. L. Minsky, *Computation: Finite and Infinite Machines*, ser. Prentice-Hall Series in Automatic Computation. Prentice-Hall, 1967.
- [30] R. Pelánek, “BEEM: Benchmarks for explicit model checkers,” in *Model Checking Software*. Springer, 2007, pp. 263–267.
- [31] T. A. Henzinger, R. Majumdar, and J.-F. Raskin, “A classification of symbolic transition systems,” *ACM Transactions on Computational Logic*, vol. 6, no. 1, pp. 1–32, 2005.
- [32] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. Thomson Course Technology, 2006.

**Theorem III.8.** *Problem III.7 is NL-hard, while the analogous decision problem for bisimulation partition is in L.*

*Proof:* We show the NL-hardness result by means of a reduction using the st-connectivity problem for leveled directed graphs. A *leveled directed graph* is a directed graph  $G$  whose nodes are partitioned in  $k > 0$  levels  $A_1, A_2, \dots, A_k$ , and every edge  $(n, m)$  of  $G$  is such that if  $n \in A_i$ , for some  $0 < i < k$ , then  $m \in A_{i+1}$ . The st-connectivity problem in a leveled directed graph asks given two nodes  $s \in A_1$  and  $t \in A_k$  whether there exists a path in  $G$  from  $s$  to  $t$ . The st-connectivity problem for leveled directed graphs is NL-complete [32, p. 333].

Now, we reduce the st-connectivity problem to our decision Problem III.7. We first observe that given an instance of the st-connectivity problem in leveled directed graphs we can add self-loops to every node of the graph while preserving the existence or not of a path from  $s$  to  $t$ . Next, given such a leveled directed graph  $G = (N, E)$  with nodes  $N = A_1 \cup A_2 \cup \dots \cup A_k$ , edges  $E$ , and two nodes  $s \in A_1$ ,  $t \in A_k$ , define the transition system  $T_s$  by taking  $G$  with set of initial states  $I = \{s\}$ . Next, consider as initial qo relation  $R_i = \{\{t\} \times N\} \cup \{(N \setminus \{t\}) \times (N \setminus \{t\})\}$ . It is easy to check, using a reasoning analogous to the undecidability proof in Section III-A, that  $R_i$  is the simulation preorder of  $T_s$  (that is,  $R_i = R_{\text{sim}}$ ). In particular, since  $G$  is a leveled directed graph, then  $t \rightarrow x$  for no state  $x$ , so that the only inclusions which need to be checked are  $R_i(N \setminus \{t\}) \subseteq \text{pre}(R_i(N \setminus \{t\}))$ ,  $R_i(N \setminus \{t\}) \subseteq \text{pre}(R_i(\{t\}))$ , and  $R_i(\{t\}) \subseteq \text{pre}(R_i(\{t\}))$ , and they all follow by definition of  $T_s$  and  $R_i$ . In turn, the simulation equivalence  $R_{\text{sim}} \cap (R_{\text{sim}})^{-1}$  induces the partition  $P_{\text{sim}} = \{N \setminus \{t\}, \{t\}\}$ . Finally, it turns out that there exists a path from  $s$  to  $t$  in  $G$  iff all the blocks of  $P_{\text{sim}}$  are reachable. More precisely,  $(N \setminus \{t\}) \cap \text{post}^*(I) \neq \emptyset$  since  $I = \{s\}$  and  $s \in N \setminus \{t\}$ ; also,  $\{t\} \cap \text{post}^*(I) \neq \emptyset$  iff there exists a path from  $s$  to  $t$ . Note that our reduction uses only two blocks as required.

Next, we show that the formulation of the decision Problem III.7 for bisimulation (rather than simulation) partition  $P_{\text{bis}}$  consisting of two blocks  $B_0, B_1$  is in L. Assume, without loss of generality, that  $I \subseteq B_0$  (the other cases are easily settled). We are left to decide whether  $B_1 \cap \text{post}^*(I) \neq \emptyset$ , which can be done by scanning one by one the transitions of the system to find a pair  $(n, m)$  with  $n \in B_0$  and  $m \in B_1$ . Indeed, assume that  $B_1 \cap \text{post}^*(I) \neq \emptyset$  holds. Therefore there exists a path from a state in  $I \subseteq B_0$  to a state in  $B_1$ . Since the bisimulation partition  $P_{\text{bis}} = \{B_0, B_1\}$ , there exists a path in  $G$  from  $I$  to  $B_1$  iff every state in  $B_0$  has a transition into  $B_1$ . Thus to check whether  $B_1 \cap \text{post}^*(I) \neq \emptyset$  holds, it suffices to find a transition  $n \rightarrow m$  with  $n \in B_0$  and  $m \in B_1$ . We need no more than logarithmic space to scan one by one the transitions and to prove the existence or absence of such an edge and we are done. ■

**Theorem IV.1 (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  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$  and  $\sigma_i \subseteq \text{post}^*(I)$ . Let  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation qo and partition w.r.t.  $R_i$ . Let  $P \triangleq \{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma} \in \text{Part}(\Sigma)$ . Then:*

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

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

*Proof:* The following facts on the output tuple  $\langle R, \sigma \rangle$  hold:

- (i) At every iteration,  $R$  is reflexive.

This holds because at the beginning  $R$  is a quasiorder and the update and refine block of  $R$  at lines 11–13 of Algorithm 1 preserves the reflexivity of  $R$ , in particular the refinement statement at line 13.

- (ii) 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)  $x \in \text{post}^*(I) \Rightarrow R(x) \cap \sigma \neq \emptyset$ .

This is proven by induction on  $n \in \mathbb{N}$  such that  $I \rightarrow^n x$ . 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  $I \rightarrow^n x' \xrightarrow{a} x$ , 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) For all  $x \in \Sigma$ ,  $R(x) \cap \sigma \neq \emptyset \Rightarrow R(x) = R_{\text{sim}}(x)$ .

Let  $\text{Sim}$  be the basic simulation algorithm as recalled in Algorithm 5. 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  $\text{Sim}$  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  $\text{Sim}$  before this iteration  $it$ . Let  $R'(y) \subseteq R(y)$  be the principal of  $R'$  used by  $\text{Sim}$  in this iteration  $it$  to refine  $R(x)$ , so that  $x \xrightarrow{a} y$ ,  $R'(x) = R(x) \not\subseteq \text{pre}_a(R'(y))$ . We have  $R(x) \cap \sigma \neq \emptyset$ ,  $x \xrightarrow{a} y$  and  $V = \emptyset$  entail  $R(x) \subseteq \text{pre}_a(R(y))$ . Hence,  $R(x) \cap \sigma \neq \emptyset$  implies  $R(y) \cap \text{post}_a(\sigma) \neq \emptyset$ , 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  $R(y) = R_{\text{sim}}(y) \subseteq R'(y) \subseteq R(y)$ , 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  $\text{Sim}$ , we have that  $R'(y) = R(y)$  holds. Thus, in both cases,  $R'(y) = R(y)$  must hold, so that  $R(x) \not\subseteq \text{pre}_a(R(y))$ . Moreover,  $R(x) \cap \sigma \neq \emptyset$ ,  $x \xrightarrow{a} y$  and  $V = \emptyset$  imply

$R(x) \subseteq \text{pre}_a(R(y))$ , which is therefore a contradiction. Thus,  $\mathbb{S} = \emptyset$ , and, in turn,  $R(x) = R_{\text{sim}}(x)$ .

(v)  $\sigma \subseteq \text{post}^*(I)$ .

This follows by  $\sigma_i \subseteq \text{post}^*(I)$  and because  $\sigma$  is updated at line 10 of Algorithm 1 by adding  $s \in I \cup \text{post}(\sigma)$  and  $\text{post}^*(I)$  is the least fixpoint of  $\lambda X. I \cup \text{post}(X)$ .

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

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

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

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

**Theorem IV.4 (Termination of Algorithm 1).** *Let  $G = (\Sigma, I, L, \rightarrow)$  with  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$  and  $\sigma_i \subseteq \text{post}^*(I)$ . Then, Algorithm 1 terminates on input  $G$ ,  $R_i$ , and  $\sigma_i$ .*

*Proof:* We first observe that each *Search* iteration adds some new state to  $\sigma$  through the update at line 10. Thus, since  $\Sigma$  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 each principal has an initial finite number of 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, so that this completes the proof. ■

**Theorem V.1 (Correctness of Algorithm 3).** *Let  $\langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \wp(\Sigma)$  be the output of Algorithm 3 on input  $G$  with  $|\Sigma| \in \mathbb{N}$ ,  $R_i \in \text{QO}(\Sigma)$ ,  $I \subseteq \sigma_i \subseteq \text{post}^*(I)$ . Let  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation qo and partition w.r.t.  $R_i$ . Let  $P \triangleq \{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma} \in$*

*Part*( $\Sigma$ ). Then:

$$\{R_{\text{sim}}(x) \mid x \in \text{post}^*(I)\} = \{R(x) \mid x \in \sigma\}, \quad (3.a)$$

$$\{B \cap \text{post}^*(I) \mid B \in P_{\text{sim}}^{\text{post}^*(I)}\} = \{B \cap \text{post}^*(I) \mid B \in P^\sigma\}, \quad (3.b)$$

$$P_{\text{sim}}^{\text{post}^*(I)} = \{P_{\text{sim}}(B) \mid B \in P^\sigma\}. \quad (3.c)$$

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

$$U = \emptyset \quad (7)$$

$$U \neq \emptyset \wedge (\text{post}(\sigma) \subseteq \sigma) \quad (8)$$

Corresponding to, respectively, the case where the algorithm executes line 25 or 30. Moreover, in the case  $U \neq \emptyset$ , we observe that  $\text{post}(\sigma) \subseteq \sigma$ , together with  $I \subseteq \sigma$  allows us to conclude that  $\text{post}^*(I) \subseteq \sigma$  by fixpoint definition of  $\text{post}^*(I)$ , and  $\text{Inv}_2$  implies  $\sigma = \text{post}^*(I)$ . Moreover, by correctness of Algorithm 2 we get that, at line 25,  $\sigma \subseteq \sigma' \subseteq \text{post}^*(I)$  and thus  $\sigma = \sigma' = \text{post}^*(I)$ .

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

(i) For all  $x \in \Sigma$ ,  $R_{\text{sim}}(x) \subseteq R(x)$ .

This holds because the *Refine* block of Algorithm 3 is always correct, so that, for all  $x \in \Sigma$ ,  $R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x)$  holds for every iteration of Algorithm 3. Moreover, correctness of Algorithm 2 ensures that line 25 preserves this property.

(ii)  $x \in \text{post}^*(I) \Rightarrow \exists s \in \sigma. R(x) = R(s)$ .

We distinguish the two possible cases at termination:

( $U = \emptyset$ ): We proceed by induction on  $n \in \mathbb{N}$  such that  $x \in \text{post}^n(I)$ . For  $n = 0$ ,  $x \in I$ , so that by initialization of  $\sigma$ ,  $x \in \sigma$ . For the inductive case we proceed by induction on  $n \in \mathbb{N}$  such that  $x \in \text{post}^n(I)$ : If  $x \in \text{post}^{n+1}(I)$  then there exists  $y \in \text{post}^n(I)$  such that  $y \rightarrow x$ , so that, by inductive hypothesis,  $R(y) = R(s')$  for some  $s' \in \sigma$ . Since  $y \rightarrow x$ ,  $V = \emptyset$  imply  $R(y) \subseteq \text{pre}(R(x))$ . Thus, since  $R$  is reflexive,  $s' \in R(s') = R(y)$  holds, implying  $s' \in \text{pre}(R(x))$ , i.e.,  $R(x) \cap \text{post}(\sigma) \neq \emptyset$ . Then,  $U = \emptyset$  implies  $\exists s \in \sigma. R(s) = R(x)$ .

( $U \neq \emptyset$ ): Since  $\sigma = \text{post}^*(I)$ , then  $x \in \text{post}^*(I) = \sigma$  and the property trivially holds.

(iii)  $(\exists s \in \sigma. R(s) = R(x)) \Rightarrow R(x) = R_{\text{sim}}(x)$ .

We distinguish the two possible cases at termination:

( $U \neq \emptyset$ ): By reflexivity,  $s \in R(s) = R(x)$  implies  $R(x) \cap \sigma \neq \emptyset$ , and thus correctness of Algorithm 2 let us conclude  $R(x) = R_{\text{sim}}(x)$ .

( $U = \emptyset$ ): Let  $\text{Sim}$  be the basic simulation algorithm taking in input a relation  $R$  and computing, for finite state systems,  $R_{\text{sim}}$ .

By (i),  $R_{\text{sim}}(z) \subseteq R(z) \subseteq R_i(z)$ , for all  $z \in \Sigma$ . Moreover,  $\text{Sim}(R) = R_{\text{sim}}$  because  $R_{\text{sim}} = \text{Sim}(R_{\text{sim}}) \subseteq \text{Sim}(R) \subseteq \text{Sim}(R_i) = R_{\text{sim}}$ . Let us assume, by contradiction, that  $\mathbb{S} \triangleq \{R(x) \mid x \in \Sigma, R(x) \neq R_{\text{sim}}(x)\} \neq \emptyset$ , so that every principal in  $\mathbb{S}$  will be refined at some iteration of  $\text{Sim}$  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  $R' \preceq R$ , 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  $x \xrightarrow{a} y$ ,  $R'(x) = R(x) \not\subseteq \text{pre}_a(R'(y))$ . We have that  $x \xrightarrow{a} y$ , along with  $\exists s \in \sigma. R(s) = R(x)$  and  $V = \emptyset$  entails  $R(x) \subseteq \text{pre}_a(R(y))$ . Hence, by reflexivity  $s \in R(s) = R(x) \subseteq \text{pre}_a(R(y))$  implies  $R(y) \cap \text{post}(\sigma) \neq \emptyset$ , so that  $U = \emptyset$  entails  $\exists s' \in \sigma. R(y) = R(s')$ .

Now, either  $R(y) \notin \mathbb{S}$ , which implies  $R(y) = R_{\text{sim}}(y)$  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  $R(x) \not\subseteq \text{pre}_a(R'(y)) = \text{pre}_a(R(y))$ , which is a contradiction to  $R(x) \subseteq \text{pre}_a(R(y))$ . Thus  $\mathbb{S} = \emptyset$  and, in turn  $R(x) = R_{\text{sim}}(x)$ .

- (iv)  $(\exists s \in \sigma. R(s) = R(x)) \Rightarrow P(x) \cap \text{post}^*(I) = P_{\text{sim}}(x) \cap \text{post}^*(I)$ .

We first show that

$$(\exists s \in \sigma. R(s) = R(x)) \Rightarrow P(x) \subseteq P_{\text{sim}}(x). \quad (9)$$

Consider an element  $y \in P(x)$ , then by definition of  $P$  it holds  $R(y) = R(x)$  and thus  $R(y) = R(s)$ , moreover, by (iii) we get  $R_{\text{sim}}(y) = R(y) = R(x) = R_{\text{sim}}(x)$  and thus by definition of  $P_{\text{sim}}$  it follows that  $y \in P_{\text{sim}}(x)$ , meaning  $P(x) \subseteq P_{\text{sim}}(x)$ , which proves (9). The inclusion  $P(x) \cap \text{post}^*(I) \subseteq P_{\text{sim}}(x) \cap \text{post}^*(I)$  follows by definition from (9). On the other hand, pick some  $y \in P_{\text{sim}}(x) \cap \text{post}^*(I)$ , then by definition of  $P_{\text{sim}}$  we get  $R_{\text{sim}}(y) = R_{\text{sim}}(x)$ , by (ii), we get that  $y \in \text{post}^*(I)$  entails  $R(y) = R(s')$  for some  $s' \in \sigma$  and (iii) implies both  $R(y) = R_{\text{sim}}(y) = R_{\text{sim}}(x) = R(x)$ . Thus by definition of  $P$ ,  $y \in P(x)$  and therefore  $P_{\text{sim}}(x) \cap \text{post}^*(I) \subseteq P(x) \cap \text{post}^*(I)$ .

- (v)  $s \in \sigma \Rightarrow P_{\text{sim}}(s) = P_{\text{sim}}(P(s))$ : Take a state  $s \in \sigma$ , then it holds that, by (9),  $s \in P(s) \subseteq P_{\text{sim}}(s)$ , and by definition of additive lifting it holds  $\bigcup_{x \in \{s\}} P_{\text{sim}}(x) \subseteq \bigcup_{x \in P(s)} P_{\text{sim}}(x) \subseteq \bigcup_{x \in P_{\text{sim}}(s)} P_{\text{sim}}(x)$  that, in turn, entails  $P_{\text{sim}}(s) \subseteq P_{\text{sim}}(P(s)) \subseteq P_{\text{sim}}(P_{\text{sim}}(s)) = P_{\text{sim}}(s)$ , and therefore  $P_{\text{sim}}(s) = P_{\text{sim}}(P(s))$ .
- (vi)  $\sigma \subseteq \text{post}^*(I)$ .

This holds since  $\sigma_i \subseteq \text{post}^*(I)$  and updates at lines 13, 27 preserve this property.

Let us now show (3.a): Consider  $x \in \text{post}^*(I)$ , then by (ii) there exists some  $s \in \sigma$  s.t.  $R(s) = R(x)$  and by (iii)  $R(x) = R_{\text{sim}}(x)$ , proving that  $R_{\text{sim}}(x) \in \{R(y) \mid y \in \sigma\}$ . On the other hand, consider a state  $s \in \sigma$ , then by (vi) we have  $s \in \text{post}^*(I)$ , moreover, by (iii) we get  $R(s) = R_{\text{sim}}(s)$  and thus we get  $R(s) \in \{R_{\text{sim}}(y) \mid y \in \text{post}^*(I)\}$ .

We now prove (3.b): Consider a block  $P_{\text{sim}}(z)$  s.t.  $P_{\text{sim}}(z) \cap \text{post}^*(I) \neq \emptyset$ , then  $P_{\text{sim}}(z) = P_{\text{sim}}(x)$  for some  $x \in \text{post}^*(I)$ . By (ii),  $R(x) = R(s)$  for some  $s \in \sigma$  and by (iii)  $R_{\text{sim}}(x) = R(x) = R(s) = R_{\text{sim}}(s)$  meaning  $P_{\text{sim}}(s) = P_{\text{sim}}(x) = P_{\text{sim}}(z)$ , moreover, by (iv) we get  $P_{\text{sim}}(s) \cap \text{post}^*(I) = P(s) \cap \text{post}^*(I)$  and by reflexivity of  $P$

we get  $P_{\text{sim}}(z) \cap \text{post}^*(I) \in \{B \cap \text{post}^*(I) \mid B \in P, B \cap \sigma \neq \emptyset\}$ .

For the other inclusion, we consider  $P(z)$  s.t.  $P(z) \cap \sigma \neq \emptyset$  and thus  $P(z) = P(s)$  for some  $s \in \sigma$  and  $R(z) = R(s)$ , by definition of  $P$ . By (iv),  $P(z) \cap \text{post}^*(I) = P_{\text{sim}}(z) \cap \text{post}^*(I)$ , moreover, (v) entails  $s \in P(z) \cap \text{post}^*(I)$  and thus  $s \in P_{\text{sim}}(z) \cap \text{post}^*(I)$ , meaning  $P_{\text{sim}}(z) \cap \text{post}^*(I) \neq \emptyset$  and thus proving  $P(z) \cap \text{post}^*(I) \in \{B \cap \text{post}^*(I) \mid B \in P_{\text{sim}}, B \cap \text{post}^*(I) \neq \emptyset\}$ .

Finally, we prove the two inclusions of (3.c):

( $\subseteq$ ): Pick a block  $P_{\text{sim}}(x)$  s.t.  $P_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset$  and consider  $z \in P_{\text{sim}}(x) \cap \text{post}^*(I)$ , then by definition of  $P_{\text{sim}}$  we have  $R_{\text{sim}}(x) = R_{\text{sim}}(z)$ , moreover by (ii) there exists some state  $s \in \sigma$  s.t.  $R(s) = R(z)$  and by (iii) we get  $R_{\text{sim}}(z) = R(z) = R(s) = R_{\text{sim}}(s)$  therefore proving  $s \in P_{\text{sim}}(z) = P_{\text{sim}}(x)$ . Moreover, by definition of  $P_{\text{sim}}$  it holds  $P_{\text{sim}}(x) = P_{\text{sim}}(s)$  and by (v) we get  $P_{\text{sim}}(x) = P_{\text{sim}}(P(s))$  thus proving the inclusion since  $s \in P(s)$ .

( $\supseteq$ ): We consider a state  $s \in \sigma$  and the corresponding  $P(s)$ , then by (v) we get  $P_{\text{sim}}(P(s)) = P_{\text{sim}}(s) \in P_{\text{sim}}$  and by (vi) we get that, since  $s \in P_{\text{sim}}(s)$ , then  $P_{\text{sim}}(s) \cap \text{post}^*(I) \neq \emptyset$ , proving the inclusion. ■

**Theorem V.3 (Termination of Algorithm 3).** *Let  $G = (\Sigma, I, L, \rightarrow)$  with  $|\Sigma| \in \mathbb{N}$ ,  $I \subseteq \sigma_i \subseteq \text{post}^*(I)$  and  $R_i \in \text{QO}(\Sigma)$ . Then, Algorithm 3 terminates on input  $G$ ,  $R_i$  and  $\sigma_i$ .*

*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  $\sigma$  through the update at line 27, or it reaches the return statement at line 25, and thus the number of *Expand* iterations is also finite. Moreover, every *Search* iteration will either add some new state to  $\sigma$  through the update at line 13, or some new state to  $U_{\text{bad}}$  through the update at line 16. Thus, since  $\Sigma$  has finitely many elements, the algorithm will always execute a finite number of *Search* iterations which expand  $\sigma$ . Finally, the number of *Search* iterations expanding  $U_{\text{bad}}$  occurring in between two iterations resetting  $U_{\text{bad}}$  to  $\emptyset$  is also finite, since  $U_{\text{bad}} \subseteq \Sigma$  and  $\Sigma$  is finite, and observing that the iterations executing  $U_{\text{bad}} := \emptyset$  are, in turn, either *Refine* iterations, non terminating *Expand* iterations, or *Search* iterations expanding  $\sigma$ , 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 25 is guaranteed to terminate by Theorem IV.4 completes the proof. ■

**Property VI.4.** *Let  $R$  be a qo and  $\langle P_R, \tau_R, Q_R \rangle$  be the induced 2PR triple. Then,  $P_R = Q_R$ .*

*Proof:* We first observe that for any qo  $R$ ,  $P_R = R \cap R^{-1}$



holds. In fact, for all  $x \in \Sigma$ , assume  $y \in P_R(x)$ , then  $R(x) = R(y)$  and by reflexivity of  $R$ ,  $xRyRx$  holds, thus proving  $(x, y), (y, x) \in R \cap R^{-1}$ . Conversely, assume  $(x, y), (y, x) \in R \cap R^{-1}$  and consider  $z \in R(x)$ , thus  $yRxRz$  and by transitivity  $z \in R(y)$ , proving  $R(x) \subseteq R(y)$ . The proof for  $R(y) \subseteq R(x)$  is symmetric.

Finally, we observe that since  $R$  is a qo, then  $R^{-1}$  is a qo too and therefore by the previous result  $Q_R = R^{-1} \cap (R^{-1})^{-1}$ . Observing that  $R^{-1} \cap (R^{-1})^{-1} = R \cap R^{-1}$  completes the proof. ■

**Property VI.2.** *If  $R \in \text{Rel}(\Sigma)$  (no assumption on  $R$ ), then  $R = R_{(P_R, \tau_R, Q_R)}$  holds.*

*Proof:* We first observe that  $\forall x \in \Sigma. R(x) = R(P_R(x))$ , by additive lifting of  $R$  over the elements of  $P_R(x)$ , and the fact that  $z \in P_R(x) \Leftrightarrow R(z) = R(x)$ . Moreover,  $P_R$  and  $Q_R$  are both partitions of  $\Sigma$ , thus for all  $x \in \Sigma$ , both  $x \in P_R(x)$  and  $x \in Q_R(y)$ . 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  $\forall z \in Q_R(y). x \in R^{-1}(y) = R^{-1}(z)$  holds, and since  $R(x) = R(P_R(x))$  then  $Q_R(y) \subseteq R(P_R(x))$ . Therefore,  $Q_R(y) \in \tau_R(P_R(x))$  holds and as a consequence  $y \in Q_R(y) \subseteq \cup \tau_R(P_R(x)) = R_{(P_R, \tau_R, Q_R)}(x)$ .

( $\supseteq$ ): Let  $y \in R_{(P_R, \tau_R, Q_R)}(x)$ , then  $Q_R(y) \in \tau_R(P_R(x))$  and thus  $Q_R(y) \subseteq R(P_R(x))$  holds. Since  $Q_R$  is a partition and  $R(P_R(x)) = R(x)$ , then we get  $y \in Q_R(y) \subseteq R(x)$ , completing the proof. ■

**Property VI.3.** *Let  $\langle P, \tau, Q \rangle$  be a 2PR triple over  $\Sigma$ . Then:*

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

*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  $x \in P(x) \subseteq \cup \tau(P(x)) = R_{\langle P, \tau, Q \rangle}(x)$ . ■

**Property VII.7.** *Let  $\langle P, \tau, Q, \sigma \rangle$  be the output of Algorithm 4 with input  $G$ ,  $R_i \in \text{QO}(\Sigma)$  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 let  $\langle P_j, \tau_j, Q_j \rangle$  be the 2PR triple after  $j = 0, \dots, t$  iterations of the algorithm, and  $R_j = R_{\langle P_j, \tau_j, Q_j \rangle}$  be the corresponding underlying relation. Then:*

$$P(x) \neq P(y) \Rightarrow R_j(x) \neq R_j(y) \text{ for some } j \in [0, t], \quad (5)$$

and a dual property for  $Q$  holds:

$$Q(x) \neq Q(y) \Rightarrow R_j^{-1}(x) \neq R_j^{-1}(y) \text{ for some } j \in [0, t]. \quad (6)$$

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

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

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 VI.4 ensures that  $P_0 = Q_0$  and thus  $Q_k(x) = Q_k(y)$  holds. For the inductive case assume  $P_k(x) = P_k(y)$  and  $Q_k(x) = Q_k(y)$ . Assume 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  $x \in P_k(x) \cap \text{pre}_a(C)$  and  $y \in P_k(x) \setminus \text{pre}_a(C)$  w.l.o.g., then by (4)  $R_{k+1}(x) \subseteq R_k(x)$ . Now since the *Refine* block does not update the underlying principal for states in  $P_k(x) \setminus \text{pre}_a(C)$ , we get  $R_k(y) = R_{k+1}(y)$  and since by hypothesis  $R_k(x) = R_k(y)$  then it follows that  $R_{k+1}(x) \subseteq R_{k+1}(y)$ , and consequently  $R_{k+1}(x) \neq R_{k+1}(y)$ , giving a contradiction to the hypothesis  $R_{k+1}(x) = R_{k+1}(y)$ , and thus proving  $P_{k+1}(x) = P_{k+1}(y)$ . We now prove the dual result on  $Q$ , assume by contradiction that  $Q_{k+1}(x) \neq Q_{k+1}(y)$ , then 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  $Q_k(x) \cap \text{pre}_a(\cup \tau_k(C)) \neq \emptyset$  and  $Q_k(x) \not\subseteq \text{pre}_a(\cup \tau_k(C))$  hold. Suppose  $Q_{k+1}(x) = Q_k(x) \cap \text{pre}_a(\cup \tau_k(C))$  (the case  $Q_{k+1}(y) = Q_k(y) \cap \text{pre}_a(\cup \tau_k(C))$  is symmetric), then  $Q_{k+1}(y) = Q_k(y) \setminus \text{pre}_a(\cup \tau_k(C))$ . Consequently, line 22 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  $b \in B'$ , then  $x \in \cup \tau_{k+1}(P_{k+1}(b))$  but  $y \notin \cup \tau_{k+1}(P_{k+1}(b))$ , which entails  $b \in R_{k+1}^{-1}(x)$  and  $b \notin R_{k+1}^{-1}(y)$ , implying  $R_{k+1}^{-1}(x) \neq R_{k+1}^{-1}(y)$ , which gives a contradiction to the hypothesis  $R_{k+1}^{-1}(x) = R_{k+1}^{-1}(y)$ , thus proving  $Q_{k+1}(x) = Q_{k+1}(y)$ . Finally, since Algorithm 4 terminates after  $t$  iterations then  $P = P_t$  and  $Q = Q_t$  and the following holds:

$$(\forall j \in [0, t]. R_j(x) = R_j(y)) \Rightarrow P(x) = P(y)$$

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

Finally, equations (5) and (6) follow from the above results taking the contrapositive. ■

## APPENDIX C

### EXTENSION TO INFINITE TRANSITION SYSTEMS

We now briefly discuss how to extend the proofs for Theorems IV.1, V.1 and VII.1 to infinite state systems. 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 IV.1, point (iii) of Theorem V.1 and point (iv) of Theorem VII.1, therefore we proceed to 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  $\text{Sim}$  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,  $R(x) \cap \sigma = \emptyset$  for Theorem IV.1,  $\nexists s \in \sigma. R(x) = R(s)$  for Theorem V.1 and  $\cup \tau(B) \cap \sigma = \emptyset$  for Theorem VII.1), and which are, therefore, not included in  $\mathbb{S}$ . However, we can use Assumption VII.2 to avoid this situation altogether since, assuming  $\mathbb{S} \neq \emptyset$  (as we do, by contradiction, in the proofs), we can pick a principal  $R(z) \in \mathbb{S}$  and some  $z' \in R(z) \setminus R_{\text{sim}}(z) \neq \emptyset$  by definition of  $\mathbb{S}$ . Assumption VII.2 therefore ensures that there exists an execution where  $\text{Sim}$  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  $\text{Sim}$  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  $\text{Inv}_1$ . The proof then proceeds as explained in the respective points (iv), (iii) and (iv) by considering the first principal in  $\mathbb{S}'$  to be refined and the corresponding iteration  $it$ , allowing us to get a contradiction which proves that  $\mathbb{S} = \emptyset$ .