

# Online Simulation Reduction

Pierre Ganty 

IMDEA Software Institute, Spain

Nicolas Manini 

IMDEA Software Institute and Universidad Politécnica de Madrid, Spain

Francesco Ranzato 

University of Padova, Italy

---

## Abstract

We study the problem of simultaneously performing reachability analysis and simulation reduction of transition systems, called online minimization by Lee and Yannakakis [1992] who settled the question for bisimulation. We call this problem *online simulation reduction* to reflect some significant dissimilarities w.r.t. Lee and Yannakakis' online bisimulation minimization. Indeed, by means of a reduction to an undecidable problem and of its relationship with graph st-connectivity, we show that by moving from bisimilarity to similarity this online reduction problem becomes fundamentally different. Then, we put forward a symbolic (i.e., processing state partitions) algorithm that performs online reduction in a complete way for the simulation quasiorder while yielding a sound over-approximation for simulation equivalence, and a second symbolic algorithm which is complete for simulation equivalence but, due to the undecidability result, reveals unavoidable limitations on its termination.

**2012 ACM Subject Classification** Theory of computation; Theory of computation → Models of computation; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Simulation, Reachability, Online Algorithm, System Reduction.

## 1 Introduction

One way to tackle the well-known state explosion problem arising in formal verification [8] is to minimize the transition system to analyze, for instance, by computing the quotient of the system w.r.t. some behavioral equivalence relation between states, typically bisimulation or simulation equivalence [9, Chapter 13]. To further reduce the size of the quotient system one can compute its reachable part only (the unreachable portion being of no/negligible interest): the returned quotiented system should include exactly the blocks that intersect the reachable states of the system. The problem of computing the quotiented system restricted to its reachable subset is commonly referred to as *online minimization* (where online refers to online algorithms that incrementally compute the reachable states).

In the early 1990s several algorithms to compute the reachable part of the bisimulation quotiented system were studied [4, 5, 23]. 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 first compute the bisimulation and next determine its reachable blocks, or, vice versa, first compute the reachable states and then compute the bisimulation restricted to the reachable part of the system.

Our starting point was the well-known online algorithm by Lee and Yannakakis [23], here referred to as the LY algorithm, which relies on a convoluted interleaving of reachability and bisimulation computation and comes with several complexity guarantees. Roughly speaking, the runtime of their algorithm depends only on the size of the output (that is, LY is output sensitive). One natural question that we set out to answer in this work is whether the LY

algorithm can be generalized to the similarity quotient (instead of the bisimulation quotient) and if so with what kind of guarantees.

The interest of similarity stems from the well-known fact that it provides a better state space reduction than bisimilarity, and the similarity quotient retains enough precision for checking all linear temporal formulas or branching temporal logic formula without quantifier switches [2, 9, 15, 16, 24]. Moreover, infinite state systems like 2D rectangular automata have infinite bisimilarity quotients, yet finite similarity quotients [19].

We focus on the problem of computing the reachable blocks of the partition induced by the simulation preorder (i.e., the greatest simulation relation). Hence, the reachable blocks of the simulation partition define the states of the reduced system. In this work we do not consider the question of computing the transitions between the blocks of this reduced systems, which is the reason why we call our problem online simulation *reduction* (as opposed to minimization). Computing transitions raises an orthogonal set of questions related to the semantics of the resulting minimized system, as discussed by Bustan and Grumberg [6] for their non-online simulation minimization algorithm. While in the bisimulation case transitions between reachable blocks defined using a  $\exists\exists$  policy (i.e.,  $B \rightarrow B'$  iff  $\exists s \in B. \exists s' \in B'. s \rightarrow s'$ ) or a  $\forall\exists$  policy (i.e.,  $B \rightarrow B'$  iff  $\forall s \in B. \exists s' \in B'. s \rightarrow s'$ ) coincide, this equivalence does not hold for the simulation case, where one can further prune the  $\forall\exists$  transitions [6].

It turns out that similarity is a preorder while bisimilarity is an equivalence. In this context, equivalences are commonly viewed and represented as state partitions whose blocks are equivalence classes. For preorder relations, the notion of “equivalence class” becomes that of *principal* of the relation: the principal of a state  $x$  is defined as the set of states greater than or equal to  $x$  for the given preorder. While the set of equivalence classes form a partition of the state space, the set of principals is not as conceptually simple, since a state can belong to more than one principal. Moreover, algorithms working with behavioral equivalences like the LY algorithm maintain, as an invariant, a partition of the state space which is incrementally refined (in their essence, they are partition refinement algorithms [18]). On the other hand, the algorithms computing the simulation preorder, such as the well-known Henzinger, Henzinger and Kopke procedure [19], refine a relation which is initially a preorder, but this property is not invariantly maintained across the refinement iterations.

In Section 3, we show, by means of a reduction to an undecidable problem for infinite state systems, and then, for finite state systems, through a reduction to the st-connectivity NL-complete problem for directed graphs, how the online reduction problem becomes fundamentally different once we move from bisimilarity to similarity. In Section 4, we put forward first an explicit algorithm and then a symbolic (i.e., processing state partitions) one, both of which perform online reduction in a sound and complete way for the simulation preorder while yielding a sound over-approximation for simulation equivalence. For finite state systems we prove the termination of both algorithms and provide examples for which the symbolic algorithm converges faster. In Section 5, we investigate a second symbolic algorithm which we prove to be complete for simulation equivalence. However, this completeness result comes at a price, since termination for finite state systems no longer holds, as an example shows.

Due to lack of space, correctness proofs of our algorithms are included in Appendix B.

**Related Work.** There is a large body of work [3, 7, 10, 12, 13, 19, 29, 30, 33] 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 [6, 15, 16]. Kučera and Mayr [21, 22] 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. No previously cited work is concerned with computing the reachable part of the simulation quotient.

The following works define algorithms computing the reachable part of the bisimulation quotient: LY algorithm [23] is still the state-of-the-art, and, before it, Bouajjani et al. [4, 5] put forward less efficient solutions. These algorithms have been revisited by Alur and Henzinger in a chapter of their unpublished book on computer-aided verification [1, Chapter 4], as well as by the theoretical and experimental comparison made by Fisler and Vardi [11]. We used their simplified presentation of the algorithms as starting point. Let us also mention that algorithms combining reachability and bisimulation computation inspired by the LY algorithm have been used for program analysis [17, 28] and hybrid systems verification [25].

To the best of our knowledge, no previous work considered the problem of *online* reduction of transition systems based on simulation relations.

## 2 Background

**Sets and Orders.** 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  $R$ ;  $\text{Rel}(\Sigma) \ni R^* \triangleq \bigcup_{n \in \mathbb{N}} R^n$  is the *reflexive-transitive closure* of  $R$ , where, inductively, for  $S \in \wp(\Sigma)$ ,  $R^0(S) \triangleq S$  and  $R^{n+1}(S) \triangleq R(R^n(S))$ . Analogously, for a function  $f : \wp(\Sigma) \rightarrow \wp(\Sigma)$ , let  $f^0(S) \triangleq S$  and  $f^{n+1}(S) \triangleq f(f^n(S))$ . If  $\mathcal{X} \in \wp(\wp(\Sigma))$  and  $\sigma \in \wp(\Sigma)$  then  $\mathcal{X}^\sigma \triangleq \{X \in \mathcal{X} \mid X \cap \sigma \neq \emptyset\}$ . A relation  $R \in \text{Rel}(\Sigma)$  is a *quasiorder* (qo, sometimes also called preorder) if it is reflexive and transitive, and  $\text{QO}(\Sigma) \triangleq \{R \in \text{Rel}(\Sigma) \mid R \text{ is a quasiorder}\}$  denotes the set of quasiorders on  $\Sigma$ . Analogously  $R \in \text{Rel}(\Sigma)$  is an *equivalence* if it is a symmetric qo. Equivalences induces partitions where a *partition* of  $\Sigma$  consists of pairwise disjoint nonempty subsets of  $\Sigma$ , called *blocks*, whose union is  $\Sigma$ . An equivalence relation of  $\text{Rel}(\Sigma)$  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 of  $s$ ,  $P(S) = \bigcup \{P(s) \in P \mid s \in S\}$ , and  $P^S = \{P(s) \in P \mid s \in S\} \in \text{Part}(P(S))$ .

A pair  $\langle P, \tau \rangle$  with  $P \in \text{Part}(\Sigma)$  and  $\tau \in \text{Rel}(P)$  is called a *partition-relation pair* (PR). A qo  $R \in \text{QO}(\Sigma)$  induces a PR  $\langle P_R, \tau_R \rangle$  defined as follows:  $\text{Part}(\Sigma) \ni P_R \triangleq \{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma}$ ;  $\text{Rel}(P) \ni \tau_R \triangleq \{(B, C) \mid C \subseteq R(B)\}$ . It turns out that if  $R$  is reflexive (resp. qo) then  $\langle P_R, \tau_R \rangle$  is a reflexive (resp. partial) order (observe that  $(B, C), (C, B) \in \tau_R \Leftrightarrow R(B) = R(C) \Leftrightarrow B = C$ ). A PR  $\langle P, \tau \rangle$  (with no hypothesis on the relation  $\tau \in \text{Rel}(P)$ ) induces a corresponding qo  $R_{\langle P, \tau \rangle} \in \text{QO}(\Sigma)$  defined by:  $R_{\langle P, \tau \rangle}(s) \triangleq \bigcup \tau^*(P(s)) = \bigcup \{C \in P \mid \exists B \in P(s). (B, C) \in \tau^*\}$ .

**Transition systems, Simulation and Bisimulation.** Let  $(\Sigma, I, \rightarrow)$  be a transition system, where  $\Sigma$  is a (possibly infinite) set of *states*,  $I \subseteq \Sigma$  is a subset of *initial states*, and  $\rightarrow \in \text{Rel}(\Sigma)$  is the *transition relation*. We define **post**:  $\wp(\Sigma) \rightarrow \wp(\Sigma)$  as the usual *successor transformer*  $\text{post}(X) \triangleq \{y \in \Sigma \mid \exists x \in X. x \rightarrow y\}$ , and, dually, **pre**:  $\wp(\Sigma) \rightarrow \wp(\Sigma)$  is the *predecessor*  $\text{pre}(X) \triangleq \{y \in \Sigma \mid \exists x \in X. y \rightarrow x\}$ . Thus,  $\text{post}^*(I) = \bigcup_{n \in \mathbb{N}} \text{post}^n(I)$  is the set of *reachable states* (from the initial states) of  $G$ .

Given a transition system  $(\Sigma, I, \rightarrow)$  and a qo  $R \in \text{QO}(\Sigma)$ , we need to be able to compute

a set of basic operations ( $\text{pre}$ ,  $\text{post}$ ,  $\cap$ ,  $\cup$ ,  $\setminus$ ), test for emptiness and inclusion, and sample a member (for nonempty sets) on the principals of  $R$ , and on every set obtained through some combination of the aforementioned operations on these principals<sup>1</sup>. We say that a transition system equipped with these operations is *effectively presented* and this notion extends to blocks, in the case of symbolic algorithms processing state partitions.

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 a labeling of a Kripke structure), a relation  $R \in \text{Rel}(\Sigma)$  is a *simulation* on  $G$  if: (1)  $R \subseteq R_i$ ; (2)  $(s, t) \in R$  and  $s \rightarrow s'$  imply  $\exists t'. t \rightarrow t'$  and  $(s', t') \in R$ . Given two principals  $R(s)$ ,  $R(s')$  such that  $s \rightarrow s'$ ,  $R(s)$  is *stable* w.r.t.  $R(s')$  when  $R(s) \subseteq \text{pre}(R(s'))$ , otherwise  $R(s)$  is called *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  $P_{\text{bis}} \in \text{Part}(\Sigma)$ .

### 3 Online Reduction for Simulation

The problem we address in this paper is the simultaneous computation of reachability analysis and simulation reduction of transition systems, that we call *online simulation reduction*, namely:

**Given:** An effectively presented transition system  $G = (\Sigma, I, \rightarrow)$  and a qo  $R_i \in \text{QO}(\Sigma)$ .

**Compute:** The reachable blocks of  $P_{\text{sim}}$ , that is  $P_{\text{sim}}^{\text{post}^*(I)}$ , where  $P_{\text{sim}}$  is the simulation partition on  $G$  w.r.t.  $R_i$ .

One first challenge we face in online reduction 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 induced by bisimulation/simulation equivalence) is naturally defined as follows:

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

When moving to simulation algorithms, a notion of reachability for principals of a relation is also needed, leading to multiple generalizations of the block-reachability notion. In fact, principal reachability 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, where, clearly, (3)  $\subseteq$  (2) and the inclusion may be strict, as shown in the following Example 3.1:

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

$$\{R(s) \in \wp(\Sigma) \mid s \in \text{post}^*(I)\} . \quad (3)$$

► **Example 3.1.** Consider the transition system  $\langle \Sigma = \{0, 1\}, I = \{1\}, \rightarrow = \{(1, 1)\} \rangle$  and the initial qo  $R_i = \Sigma \times \Sigma$ . The resulting simulation quasiorder is given by  $R_{\text{sim}}(0) = \{0, 1\}$ ,  $R_{\text{sim}}(1) = \{1\}$ , and 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  $R_{\text{sim}}(1)$ . ◀

<sup>1</sup> The required combinations of operations is specific to each algorithm and will become clear from the algorithm pseudocode.

Let us remark that it might also happen that, for some systems, the reachable principals according to (2) are infinitely many, while for (3) are finitely many.

### 3.1 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 turns out to be, in general, unsolvable even under the assumption that  $P_{\text{sim}}$  is a finite partition. This negative result is in stark contrast with the problem of computing reachable blocks for the bisimulation partition  $P_{\text{bis}}$ . In fact, Lee and Yannakakis' minimization algorithm for bisimulation equivalence always terminates when  $P_{\text{bis}}$  is finite [23, Theorem 3.1 and following paragraph]. In order to prove our negative result, we show that the 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 in  $\mathbb{N}$ ,  $H \subseteq Q$  is a set of halting states, 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  $c$ : these 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  $(q, j_1, \dots, j_m)$  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  $c_1, \dots, c_n$  such that:  $c_1 \rightarrow \dots \rightarrow c_n$ ;  $c_1 = (q_0, j_1, \dots, j_{|C|})$ ; and  $c_n$  is halting.

The following *halting problem for 2-CMs* is known to be undecidable [26]:

**Given:** A 2-CM  $M$ .

**Decide:** Can  $M$  halt on input  $(0, 0)$ ?

We will show that this decision problem can be reduced to the following *reachability problem for simulation equivalence*, which, therefore, turns out to be undecidable:

**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  $\langle \Sigma = Q \times \mathbb{N}^2, I = \{(q_0, 0, 0)\}, \rightarrow \rangle$  generated by  $M$  w.r.t. an initial go relation on  $Q \times \mathbb{N}^2$ .

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

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  $I \triangleq \{(q_0, 0, 0)\}$ .

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  $R(c_y) = Q \times \mathbb{N}^2$ , 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  $c_x, c_y$  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, it turns out that  $R_i$  is the simulation qo  $R_{\text{sim}}$  of  $M$ , that is:

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

In turn, the simulation equivalence  $R_{\text{sim}} \cap (R_{\text{sim}})^{-1}$  induces the following finite simulation partition:

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

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

► **Theorem 3.2.** *The reachability problem for simulation equivalence of 2-CMs 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.

### 3.2 Complexity for Finite State Systems

We now 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, by checking  $\text{post}^*(I) \cap B = ? \emptyset$ , for any block  $B$  in  $P_{\text{bis}}$  or  $P_{\text{sim}}$ , we solve the reachability problem for  $B$ .

For the case of bisimulation, deciding the reachability of  $B \in P_{\text{bis}}$  can be easily solved in  $O(|P_{\text{bis}}^{\text{post}^*(I)}|)$  time by leveraging the definition of bisimulation: picking any 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 any  $B \in P_{\text{bis}}$ ,  $x$  can reach  $B$  iff  $y$  can reach  $B$ , so that it is enough to pick one state per block of  $P_{\text{bis}}$ .

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 3.3.** Consider the following transition system  $\mathcal{S}_1$ , where 1 is the initial state.



Here, we have that  $P_{\text{sim}}$  w.r.t.  $R_i = \Sigma \times \Sigma$  is given by the blocks 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 the block  $[0, 0] \in P_{\text{sim}}$  is actually reachable.

Let us remove the transition  $n \rightarrow 0$ , i.e., consider the following transition system  $\mathcal{S}_2$ .



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  $\mathcal{S}_1$  the state 0 is reachable, while in  $\mathcal{S}_2$  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  $\mathcal{S}_1$ , while  $P_{\text{bis}} = P_{\text{sim}}$  for  $\mathcal{S}_2$ . ◀

Let us now turn the previous example into a formal complexity argument by showing that the following *reachability problem for simulation equivalence on finite transition systems* is hard for NL (the nondeterministic logarithmic space complexity class):

**Given:** A finite transition system  $G = (\Sigma, I, \rightarrow)$  and the simulation partition  $P_{\text{sim}}$  of  $G$  w.r.t. an initial qo relation.

**Decide:** Does  $\forall B \in P_{\text{sim}}: \text{post}^*(I) \cap B \neq \emptyset$  hold?

We show the NL-hardness result by means of a reduction from 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 known to be NL-complete [32, p. 333].

Now, we will reduce the st-connectivity problem to our problem of deciding whether all the blocks of the simulation partition are reachable. 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  $I \triangleq \{s\}$ . Next, consider as initial qo relation  $R_i \triangleq \{\{t\} \times N\} \cup \{(N \setminus \{t\}) \times (N \setminus \{t\})\}$ . It is easy to check, using a reasoning analogous to the undecidability proof in Section 3.1, 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.

Next, we show that the reachability problem for bisimulation equivalence on finite transition systems with two equivalence classes is in L (the deterministic logarithmic space complexity class). In input we have a finite transition system and the two blocks  $B_0, B_1$  of the bisimulation partition  $P_{\text{bis}}$ . 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  $n \in I$  and  $m \in B_1$ . Since  $P_{\text{bis}}$  is the bisimulation partition, it turns out that if there exists such a transition  $(n, m) \in B_0 \times B_1$  then every state of  $B_0$  has a transition into  $B_1$ . 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. We have thus proved the following result.

► **Theorem 3.4.** *On finite transition systems, the reachability problem for simulation equivalence is NL-hard, while for bisimulation equivalence is in L.*

## 4 A Sound Online Reduction Algorithm

We put forward a first algorithm which, given a finite transition system  $G$  and an initial qo  $R_i$ , computes the reachable principals of  $R_{\text{sim}}$  according to (2), along with a sound overapproximation of the blocks of  $P_{\text{sim}}^{\text{post}^*(I)}$ . This algorithm is explicit, meaning that it maintains an explicit representation of a relation  $R$  between states through its principals  $R(x)$  for any state  $x$ .

### Algorithm 1 Explicit Sound Algorithm

---

**Input:** A finite transition system  $G = (\Sigma, I, \rightarrow)$  and an initial qo  $R_i \in \text{QO}(\Sigma)$

```

1 forall  $x \in \Sigma$  do  $\wp(\Sigma) \ni R(x) := R_i(x)$ ;
2  $\wp(\Sigma) \ni \sigma := \emptyset$ ;
3 while true do
  // Inv1:  $(\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x))$ 
  // Inv2:  $(\sigma \subseteq \text{post}^*(I))$ 
  // Inv3:  $(\forall x \in \Sigma. x \in R(x))$ 
4    $U := \{x \in \Sigma \mid R(x) \cap \sigma = \emptyset, R(x) \cap (I \cup \text{post}(\sigma)) \neq \emptyset\}$ ;
5    $V := \{\langle x, x' \rangle \in \Sigma^2 \mid R(x) \cap \sigma \neq \emptyset, x \rightarrow x', R(x) \not\subseteq \text{pre}(R(x'))\}$ ;
6   nif
7      $(U \neq \emptyset) \rightarrow \text{Search} :$ 
8     choose  $x \in U$ ;
9     choose  $s \in R(x) \cap (I \cup \text{post}(\sigma))$ ;
10     $\sigma := \sigma \cup \{s\}$ ;
11     $(V \neq \emptyset) \rightarrow \text{Refine} :$ 
12    choose  $\langle x, x' \rangle \in V$ ;
13     $R(x) := R(x) \cap \text{pre}(R(x'))$ ;
14     $(U = \emptyset \wedge V = \emptyset) \rightarrow \text{return } \langle R, \sigma \rangle$ ;
15  fin
16 end

```

---

Algorithm 1 follows the pattern of the online bisimulation minimization algorithm given by Alur and Henzinger [1, Algorithm 4.4]. This algorithm explicitly maintains a current relation  $R \in \text{Rel}(\Sigma)$  represented through its principals  $R(x) \in \wp(\Sigma)$ , and a set  $\sigma \in \wp(\Sigma)$  containing



states reachable from  $I$ , at most one state per principal, 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 pair  $\langle x, x' \rangle$  belongs to  $V$  if  $R(x)$  is known to be reachable and it can be refined by  $R(x')$ , i.e.,  $x \rightarrow x'$  and  $R(x) \not\subseteq \text{pre}(R(x'))$ . The algorithm either updates the reachability information for some principal in  $U$  or stabilizes some pair  $\langle x, x' \rangle \in V$  by refining  $R(x)$ . In our pseudocode we use a **nif** statement which is a *nondeterministic choice* between guarded commands. In Algorithm 1 we have three guarded commands: either the *Search* (lines 7–10) or the *Refine* blocks (lines 11–13) are executed, or when both their guards are false, the return statement is taken. Therefore, every execution consists of an arbitrary interleaving of *Search* and *Refine* possibly followed by the return at line 14 (when the algorithm terminates). 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,  $R^\sigma$  precisely contains the reachable principals of  $R_{\text{sim}}$ , where the relation  $R$  may well have unstable principals, so that  $R$  need not to be equal to  $R_{\text{sim}}$ .

Furthermore, it turns out that Algorithm 1 yields a sound overapproximation of  $P_{\text{sim}}^{\text{post}^*(I)}$ .

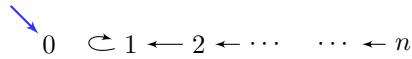
► **Theorem 4.1** (Correctness of Algorithm 1). *Let  $\langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \wp(\Sigma)$  be the output of Algorithm 1 on input  $R_i \in \text{QO}(\Sigma)$  where  $\Sigma$  is finite, and let  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  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:*

$$\{R_{\text{sim}}(x) \mid x \in \Sigma, R_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset\} = \{R(x) \mid x \in \Sigma, R(x) \cap \sigma \neq \emptyset\}, \quad (4)$$

$$\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\} \subseteq \{B \in P \mid B \cap \sigma \neq \emptyset\}. \quad (5)$$

The explicit representation used in Algorithm 1 comes with some drawbacks, shown by the following example.

► **Example 4.2.** Consider the following finite state system with  $\Sigma = [0, n]$ ,  $I = \{0\}$ , initial relation  $R_i = \Sigma \times \Sigma$  and the following transition relation:



We have that  $R_{\text{sim}}(0) = [0, n]$ , and, for all  $k \in [1, n]$ ,  $R_{\text{sim}}(k) = [1, n]$ . Algorithm 1 first computes  $\sigma = \{0\}$ , hence  $V = \{\langle 1, 1 \rangle, \langle 2, 1 \rangle, \langle 3, 2 \rangle, \dots, \langle n-1, n \rangle\}$ , so that each principal  $R(k)$ , for  $k \in [1, n]$  needs to be refined:  $R(1) := R(1) \cap \text{pre}(R(1)) = [1, n]$ , and, for  $k \geq 2$ ,  $R(k) := R(k) \cap \text{pre}(R(k-1)) = [1, n]$ . ◀

Example 4.2 shows that by relying on an explicit representation for the principals of the relation  $R$ , the algorithm could perform  $O(|\Sigma|)$  refinements of principals, although all of them will be refined in the same way. In order to address this problem, we put forward a

revision of Algorithm 1 based on a symbolic representation of  $R$  through a PR pair.

■ **Algorithm 2** Sound Symbolic Algorithm

---

**Input:** An effectively presented trans. sys.  $G = (\Sigma, I, \rightarrow)$  and qo  $R_i \in \text{QO}(\Sigma)$

```

1 Part( $\Sigma$ )  $\ni P := \{y \in \Sigma \mid R_i(y) = R_i(x)\}_{x \in \Sigma}$ ;
2 forall  $B \in P$  do  $\wp(P) \ni \tau(B) := \{C \in P \mid C \subseteq R_i(B)\}$ ;
3  $\wp(\Sigma) \ni \sigma := \emptyset$ ;
4 while true do
    // Inv1:  $(\forall x. R_{\text{sim}}(x) \subseteq \cup \tau(P(x)) \subseteq R_i(x))$ 
    // Inv2:  $(\sigma \subseteq \text{post}^*(I))$ 
    // Inv3:  $(\forall B \in P. B \in \tau(B))$ 
5  $U := \{B \in P \mid \cup \tau(B) \cap \sigma = \emptyset, \cup \tau(B) \cap (I \cup \text{post}(\sigma)) \neq \emptyset\}$ ;
6  $V := \{(B, C) \in P^2 \mid \cup \tau(B) \cap \sigma \neq \emptyset, B \cap \text{pre}(C) \neq \emptyset, \cup \tau(B) \not\subseteq \text{pre}(\cup \tau(C))\}$ ;
7 nif
8    $(U \neq \emptyset) \rightarrow \text{Search} :$ 
9     choose  $B \in U$ ;
10    choose  $s \in \cup \tau(B) \cap (I \cup \text{post}(\sigma))$ ;
11     $\sigma := \sigma \cup \{s\}$ ;
12    $(V \neq \emptyset) \rightarrow \text{Refine} :$ 
13     choose  $(B, C) \in V$ ;
14      $S := \text{pre}(\cup \tau(C))$ ;
15     // Refine the partition  $P$ , it could happen that  $P$  does not change
16      $P.\text{replace}(B, \{B \cap S, B \setminus S\})$ ;
17     // Update the relation  $\tau$  by replacing  $B$  with  $\{B \cap S, B \setminus S\}$  everywhere
18      $\tau(B \setminus S) := \tau(B)$ ;  $\tau(B \cap S) := \tau(B)$ ;
19     forall  $D \in P$  do  $\tau(D).\text{replace}(B, \{B \cap S, B \setminus S\})$ ;
20     // Refine  $\tau(B \cap S)$ : this refinement surely happens
21      $\tau(B \cap S) := \{E \in \tau(B \cap S) \mid E \subseteq S\}$ ;
22    $(U = \emptyset \wedge V = \emptyset) \rightarrow \text{return } \langle P, \tau, \sigma \rangle$ ;
23 fin
24 end

```

---

Algorithm 2 is *symbolic* in the sense that it symbolically represents and processes state relations as partition-relation pairs  $\langle P, \tau \rangle$ , where  $\tau$  is a reflexive relation between blocks of  $P$ , so that, for each state  $x$ ,  $\cup \tau(P(x))$  is a set of states candidates to simulate  $x$ , and all the states in the same block  $P(x)$  are candidates to be simulation equivalent. Such symbolic representation for simulation algorithms is known to be remarkably 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 [19, Sections 2.2 and 3], and has been shown to be advantageous in terms of space and time efficiency also for finite state systems [7, 30].

This second algorithm is designed as a symbolic counterpart of Algorithm 1. Here,  $R^\sigma \triangleq \{\cup \tau(B) \mid B \in P, \cup \tau(B) \cap \sigma \neq \emptyset\}$  includes the “symbolic principals”  $\cup \tau(B)$  which are already known to contain a reachable state. Also, the set  $V$  contains unstable pairs of blocks: a pair  $(B, C)$  is in  $V$  if  $\cup \tau(B)$  is known to be reachable and can be refined by  $\cup \tau(C)$ , i.e.,  $B \cap \text{pre}(C) \neq \emptyset$  and  $\cup \tau(B) \not\subseteq \text{pre}(\cup \tau(C))$ . The *Refine* block stabilizes a pair  $(B, C) \in V$  by possibly splitting the block  $B$  into  $B \cap \text{pre}(\cup \tau(C))$  and  $B \setminus \text{pre}(\cup \tau(C))$  (lines 14–17), and then by refining the symbolic principal  $\cup \tau(B \cap \text{pre}(\cup \tau(C)))$  by removing all the blocks not contained into  $\text{pre}(\cup \tau(C))$  (line 18). Upon termination,  $R^\sigma$  precisely contains the reachable

symbolic principals of  $R_{\text{sim}}$ , while the sound overapproximation of  $P_{\text{sim}}^{\text{post}^*(I)}$  given by (5) of Theorem 4.1 for the explicit algorithm is, this time, guaranteed by the output partition  $P$  of Algorithm 2.

► **Theorem 4.3** (Correctness of Algorithm 2). *Let  $\langle P, \tau, \sigma \rangle \in \text{Part}(\Sigma) \times \text{Rel}(P) \times \wp(\Sigma)$  be the output of Algorithm 2 on input  $R_i \in \text{QO}(\Sigma)$ , and 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.  $R_i$ , where  $P_{\text{sim}}$  is assumed to be a recursively enumerable set. Then:*

$$\{R_{\text{sim}}(x) \mid x \in \Sigma, R_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset\} = \{\cup \tau(P(x)) \mid x \in \Sigma, \cup \tau(P(x)) \cap \sigma \neq \emptyset\}, \quad (6)$$

$$\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\} \subseteq \{B \in P \mid \cup \tau(B) \cap \sigma \neq \emptyset\}. \quad (7)$$

Let us point out that the assumption of recursive enumerability for the simulation partition  $P_{\text{sim}}$  (notice that if the state space  $\Sigma$  is r.e. then  $P_{\text{sim}}$  is r.e. as well) will be used in the proof of Theorem 4.3 (in Appendix B) to guarantee that a symbolic simulation algorithm computing  $P_{\text{sim}}$ , such as those in [7, 19, 30], will eventually compute all the blocks in  $P_{\text{sim}}$ , even if it will run forever when  $P_{\text{sim}}$  is an infinite set.

► **Example 4.4** (Example 4.2 continued). By running Algorithm 2 on Example 4.2 we get  $\sigma = \{0\}$ , so that the initial partition  $P = \{[0, n]\}$ , with  $\tau([0, n]) = \{[0, n]\}$ , is split into  $P = \{[0, 0], [1, n]\}$  because  $\cup \tau([0, n]) \cap \sigma \neq \emptyset$ ,  $[0, n] \cap \text{pre}([0, n]) \neq \emptyset$  and  $S = \text{pre}(\cup \tau([0, n])) = [1, n]$  splits the block  $[0, n]$  into  $[0, 0]$  and  $[1, n]$  and, in turn, updates the relation  $\tau$  so as that  $\tau([0, 0]) = \{[0, 0], [1, n]\}$  and  $\tau([1, n]) = \{[1, n]\}$ . After this iteration, it turns out that  $U = \emptyset$  and  $V = \emptyset$ , because  $\cup \tau([1, n]) \cap \sigma \neq \emptyset$ ,  $[1, n] \cap \text{pre}([1, n]) \neq \emptyset$  but  $\cup \tau([1, n]) \subseteq \text{pre}(\cup \tau([1, n]))$ . Thus, Algorithm 2 terminates just after two iterations, one computing  $\sigma = \{0\}$ , and a second iteration for splitting  $P$  and refining  $\tau$ , as compared to the  $n$  iterations needed by the explicit Algorithm 1 to converge. ◀

The following example shows that the containment (7) of Theorem 4.3 may be strict.

► **Example 4.5.** Let us consider the system  $\langle \Sigma = \{1, 2\}, I = \{1\}, \rightarrow = \{(1, 1)\} \rangle$ , with  $R_i = \{1, 2\} \times \{1, 2\}$ . 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 2 on input  $R_i$  gives in output  $P = P_{\text{sim}}$ ,  $\tau = \{([1, 1], [1, 1]), ([2, 2], [1, 1]), ([2, 2], [2, 2])\}$ ,  $\sigma = \{1\}$ . Thus, the containment (7) turns out to be strict:  $\{[1, 1]\} \subsetneq \{[1, 1], [2, 2]\}$ . ◀

It turns out that both Algorithms 1 and 2 always terminate on finite state systems.

► **Theorem 4.6** (Termination for Finite Systems). *Let  $G = (\Sigma, I, \rightarrow)$  be a finite state system (i.e.,  $|\Sigma| \in \mathbb{N}$ ), and  $R_i \in \text{QO}(\Sigma)$ . Then, Algorithms 1 and 2 terminate on input  $G$ ,  $R_i$ .*

**Proof.** We first observe that any *Search* iteration will surely add some new state to  $\sigma$  through the update at line 11 (resp., 10). Thus, since  $\Sigma$  has finitely many elements, both Algorithms 1 and 2 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 (possibly more than one in the symbolic Algorithm 2) of the current relation (implicitly encoded by  $P, \tau$  in Algorithm 2), 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*, both algorithms terminate after a finite number of iterations. Observing that each iteration computes a finite number of operations (in particular, lines 16–18 of Algorithm 2 iterate over the blocks of  $P$ , which are finitely many because  $P$  is a partition of  $\Sigma$ ) completes the proof. ◀

## 5 A Complete Online Reduction Algorithm

In the following we introduce Algorithm 3, a variation of Algorithm 2 which achieves completeness in the characterization of the reachable blocks of  $P_{\text{sim}}$ . However, as we will see, we pay a high price for getting completeness as Algorithm 3 is not even guaranteed to terminate on finite state systems (cf. Examples 5.3 and 5.4). This is in stark contrast compared to the Algorithm 2 whose convergence is guaranteed on finite state systems, as shown in Theorem 4.6. Nontermination is expected for infinite state systems due to the undecidability of the reachability problem for  $P_{\text{sim}}$  shown in Section 3.1. Thus, in general, Algorithm 3 does not terminate even when  $P_{\text{sim}}$  turns out to be a finite partition.

In order to achieve completeness, we put forward a slight but key modification of Algorithm 2, where we will consider as reachable the blocks of the current partition rather than the principals of the current partition-relation pair. In fact, we modify the definitions of  $U$  and  $V$  at lines 5 and 6, where the unreachability/reachability in  $U$  and  $V$  of Algorithm 2 for a principal  $\cup\tau(B)$  through witnesses in  $\sigma$ , i.e. whether  $\cup\tau(B) \cap \sigma$  is empty/nonempty, is replaced in Algorithm 3 by the unreachability/reachability of the single block  $B$ , i.e. whether  $B \cap \sigma$  is empty/nonempty.

### Algorithm 3 Complete Symbolic Algorithm

---

**Input:** A transition system  $G = (\Sigma, I, \rightarrow)$  and an initial qo  $R_i \in \text{QO}(\Sigma)$

```

1 Part( $\Sigma$ )  $\ni P := \{y \in \Sigma \mid R_i(y) = R_i(x)\}_{x \in \Sigma}$ ;
2 forall  $B \in P$  do  $\wp(P) \ni \tau(B) := \{C \in P \mid C \subseteq R_i(B)\}$ ;
3  $\wp(\Sigma) \ni \sigma := \emptyset$ ;
4 while true do
    // Inv1:  $(\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq \cup\tau(P(x)) \subseteq R_i(x))$ 
    // Inv2:  $(\sigma \subseteq \text{post}^*(I))$ 
    // Inv3:  $(\forall B \in P: B \in \tau(B))$ 
5  $U := \{B \in P \mid B \cap \sigma = \emptyset, \cup\tau(B) \cap (I \cup \text{post}(\sigma)) \neq \emptyset\}$ ;
6  $V := \{(B, C) \in P^2 \mid B \cap \sigma \neq \emptyset, B \cap \text{pre}(C) \neq \emptyset, \cup\tau(B) \not\subseteq \text{pre}(\cup\tau(C))\}$ ;
7 nif
8    $(U \neq \emptyset) \rightarrow \text{Search} :$ 
9     choose  $B \in U$ ;
10    choose  $s \in \cup\tau(B) \cap (I \cup \text{post}(\sigma))$ ;
11     $\sigma := \sigma \cup \{s\}$ ;
12    $(V \neq \emptyset) \rightarrow \text{Refine} :$ 
13     choose  $(B, C) \in V$   $S := \text{pre}(\cup\tau(C))$ ;
14      $P.\text{replace}(B, \{B \cap S, B \setminus S\})$ ;
15      $\tau(B \setminus S) := \tau(B)$ ;  $\tau(B \cap S) := \tau(B)$ ;
16     forall  $D \in P$  do  $\tau(D).\text{replace}(B, \{B \cap S, B \setminus S\})$ ;
17      $\tau(B \cap S) := \{E \in \tau(B \cap S) \mid E \subseteq S\}$ ;
18    $(U = \emptyset \wedge V = \emptyset) \rightarrow \text{return } \langle P, \tau, \sigma \rangle$ ;
19 fin
20 end
```

---

The main feature of Algorithm 3 is that, when it terminates with output  $\langle P, \tau, \sigma \rangle$ , the blocks of  $P$  containing exactly one state witness of reachability in  $\sigma$  coincide with the reachable blocks of  $P_{\text{sim}}$ . On the other hand, Algorithm 3 is also complete for the reachable principals of the simulation qo  $R_{\text{sim}}$  where reachability is given by (3).

► **Theorem 5.1** (Correctness of Algorithm 3). *Let  $\langle P, \tau, \sigma \rangle \in \text{Part}(\Sigma) \times \text{Rel}(\wp(\Sigma)) \times \wp(\Sigma)$  be the output of Algorithm 3 on input  $R_i \in \text{QO}(\Sigma)$ , and  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation qo and partition w.r.t.  $R_i$ , where  $P_{\text{sim}}$  is assumed to be a recursively enumerable set. Then:*

$$\{R_{\text{sim}}(x) \mid x \in \text{post}^*(I)\} = \{\cup \tau(P(x)) \mid x \in \sigma\}, \quad (8)$$

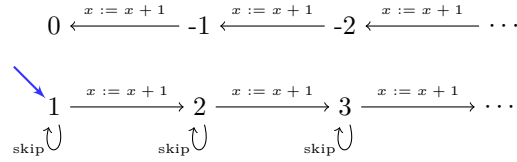
$$\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\} = \{B \in P \mid B \cap \sigma \neq \emptyset\}. \quad (9)$$

The following example shows that Algorithm 3 may terminate on infinite state systems despite the divergence of both reachability analysis and simulation quasiorder computation.

► **Example 5.2.** Consider the following nondeterministic program:

**while** ( $x \neq 0$ ) **do** (**if**  $x < 0$  **then**  $x := x + 1$  **else** (**if**  $(*)$  **then**  $x := x + 1$  **else** skip))

and the corresponding infinite state system:



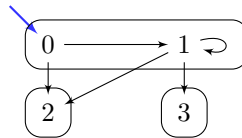
Hence, each node represents the value stored by the integer variable  $x$  which can be updated by transitions labeled by  $x := x + 1$  while skip is a no-op.

Take as initial states  $I = \{1\}$  and as initial qo the following relation: for all  $k \leq 0$ ,  $R_i(k) = \mathbb{Z}$ , and, for all  $k \geq 1$ ,  $R_i(k) = [1, +\infty[$ , so that the initial partition of Algorithm 3 is  $P \triangleq \{-\infty, 0], [1, +\infty[ \in \text{Part}(\mathbb{Z})$  with  $\tau(-\infty, 0] = \{-\infty, 0], [1, +\infty[$  and  $\tau([1, +\infty[) = \{[1, +\infty[$ . Computing  $\text{post}^*(I) = [1, +\infty[$  does not converge in finite time and neither does any simulation algorithm aiming at computing  $R_{\text{sim}}$  because for all  $k \geq 1$ ,  $R_{\text{sim}}(k) = [1, +\infty[$ , and, for all  $k \leq 0$ ,  $R_{\text{sim}}(k) = ]-\infty, k]$ , so that the simulation partition is infinite:  $P_{\text{sim}} = \{[k, k] \mid k \leq 0\} \cup \{[1, +\infty[$ .

By contrast, Algorithm 3 terminates with  $\sigma = \{1\}$ , with the initial  $P$  and  $\tau$  above: this because after *Search* computed  $\sigma = \{1\}$ , the block  $[1, +\infty[$  becomes reachable and the block  $]-\infty, 0]$  is unreachable (even though its principal  $\cup \tau(-\infty, 0] = \mathbb{Z}$  contains a reachable state in  $\sigma$ ); moreover, the pair  $([1, +\infty[, [1, +\infty[)$  is not in  $V$  because it is stable; hence, both  $U$  and  $V$  are empty, and Algorithm 3 immediately terminates. ◀

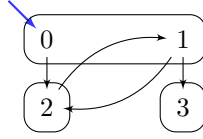
The following examples exhibit some limitations of Algorithm 3 w.r.t. nontermination on finite state systems. These divergent executions constitute a major difference w.r.t. Algorithm 2, which, as shown in Theorem 4.6, always terminates on such systems.

► **Example 5.3.** Consider the following finite state system, where 0 is the initial state, and an initial qo defined as  $R_i(0) = R_i(1) = \{0, 1\}$ ,  $R_i(2) = \{2\}$  and  $R_i(3) = \{2, 3\}$ , so that the initial PR pair in the execution of Algorithm 3 is  $P_i = \{[0, 1], [2, 2], [3, 3]\}$ ,  $\tau_i([0, 1]) = \{[0, 1]\}$ ,  $\tau_i([2, 2]) = \{[2, 2]\}$  and  $\tau_i([3, 3]) = \{[2, 2], [3, 3]\}$ . It turns out that the initial relation  $R_i$  coincides with the simulation quasiorder, i.e.,  $R_{\text{sim}} = R_i$ , hence  $P_{\text{sim}} = P_i$  the block of which are depicted below.



After one iteration of Algorithm 3, it holds  $\sigma = \{0\}$ , and at the following iteration we get  $U = \{[2, 2], [3, 3]\}$ , so that choosing any of the two blocks leads to  $\sigma = \{0, 2\}$  (note that  $V \neq \emptyset$  will never occur since  $R_i = R_{\text{sim}}$ ). At this point,  $U = \{[3, 3]\}$  since  $[3, 3] \cap \sigma = \emptyset$  and  $\cup\tau([3, 3]) \cap \text{post}(\sigma) = \{2\} \neq \emptyset$ . Then, executing *Search* adds no new element to  $\sigma$ , so that Algorithm 3 will diverge by executing the *Search* block infinitely often, without updating any variable, thus entailing that  $U = \emptyset$  will never occur. ◀

► **Example 5.4.** Consider the following finite state system, where 0 is the initial state, and the initial qo  $R_i$  (and PR pair in the execution of Algorithm 3) from Example 5.3. We have that  $R_i$  coincides with the simulation quasiorder, i.e.,  $R_{\text{sim}} = R_i$ .



After two iterations of Algorithm 3, it holds  $\sigma = \{0, 2\}$  (note that  $V \neq \emptyset$  will never occur since  $R_i = R_{\text{sim}}$ ). At this point,  $U = \{[3, 3]\}$  since  $[3, 3] \cap \sigma = \emptyset$  and  $\cup\tau([3, 3]) \cap \text{post}(\sigma) = \{2\} \neq \emptyset$ . Then, performing a *Search* will not add any new element to  $\sigma$ , so that Algorithm 3 will diverge by executing the *Search* block infinitely often, without updating any variable, thus entailing that  $U = \emptyset$  will never occur.

Notice that, contrary to Example 5.3, in this transition system, node 1 is not reachable from 0 by any path fully contained in  $P_{\text{sim}}(0)$ . ◀

## 6 Conclusion and Future Work

We presented and proved soundness of several algorithms with various tradeoffs, assumptions and limitations for the online simulation reduction problem, which discloses some fundamental differences in decidability and complexity w.r.t. to the case of bisimulation studied by Lee and Yannakakis [23]. To the best of our knowledge, this is the first investigation of this problem. Algorithm 2 is the most relevant one for practical purposes since it converges on all finite state systems and on some (but not all, due to an undecidability result which we have shown) infinite state systems.

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

- To tackle the minimization problem, that is, study the various definitions of edges between abstract states and how they affect the semantics of the resulting system. To this aim, a starting point will be the non-online simulation minimization algorithm by Bustan and Grumberg [6].
- To carry out an implementation of Algorithms 2 and 3, leveraging the data structures such as queue and stacks used by Lee and Yannakakis' online algorithm [23] for bisimulation.
- To establish runtime guarantees of the algorithms as the ones stated by Lee and Yannakakis [23, Section 3].
- To investigate whether online minimization algorithms can be designed for further behavioral relations. One promising relation is branching bisimilarity for labeled transition systems or stuttering equivalence for Kripke structures, which have been recently [14, 20] proven to be symbolically computable in  $O(|\rightarrow| \log |\Sigma|)$  time, the same complexity of the renowned Paige-Tarjan bisimulation algorithm [27].



## References

- 1 Rajeev Alur and Thomas A. Henzinger. Computer-Aided Verification. Chapter 4: Graph Minimization, October 1999. URL: <https://www.cis.upenn.edu/~alur/CAVBook.pdf>.
- 2 Saddek Bensalem, Ahmed Bouajjani, Claire Loiseaux, and Joseph Sifakis. Property preserving simulations. In *Proceedings of the Fourth International Workshop on Computer Aided Verification, CAV '92*, volume 663 of *Lecture Notes in Computer Science*, pages 260–273. Springer, 1992. doi:10.1007/3-540-56496-9\_21.
- 3 Bard Bloom and Robert Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Sci. Comput. Program.*, 24(3):189–220, 1995. doi:10.1016/0167-6423(95)00003-B.
- 4 A. Bouajjani, J-C. Fernandez, and N. Halbwachs. Minimal model generation. In Edmund M. Clarke and Robert P. Kurshan, editors, *Computer-Aided Verification*, Lecture Notes in Computer Science, pages 197–203, Berlin, Heidelberg, 1991. Springer. doi:10.1007/BFb0023733.
- 5 A. Bouajjani, J. C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Science of Computer Programming*, 18(3):247–269, June 1992. doi:10.1016/0167-6423(92)90018-7.
- 6 Doron Bustan and Orna Grumberg. Simulation-based minimization. *ACM Trans. Comput. Log.*, 4(2):181–206, 2003. doi:10.1145/635499.635502.
- 7 Gérard Cécé. Foundation for a series of efficient simulation algorithms. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005069.
- 8 Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Progress on the state explosion problem in model checking. In *Informatics - 10 Years Back. 10 Years Ahead*, volume 2000 of *Lecture Notes in Computer Science*, pages 176–194. Springer, 2001. doi:10.1007/3-540-44577-3\_12.
- 9 Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of Model Checking*. Springer, 1st edition, 2018.
- 10 Silvia Crafa, Francesco Ranzato, and Francesco Tapparo. Saving space in a time efficient simulation algorithm. *Fundam. Informaticae*, 108(1-2):23–42, 2011. doi:10.3233/FI-2011-412.
- 11 Kathi Fisler and Moshe Y. Vardi. Bisimulation minimization and symbolic model checking. *Formal Methods Syst. Des.*, 21(1):39–78, 2002. doi:10.1023/A:1016091902809.
- 12 Raffaella Gentilini, Carla Piazza, and Alberto Policriti. From bisimulation to simulation: Coarsest partition problems. *Journal of Automated Reasoning*, 31(1):73–103, 2003. doi:10.1023/A:1027328830731.
- 13 Rob van Glabbeek and Bas Ploeger. Correcting a space-efficient simulation algorithm. In *Proceedings of the International Conference on Computer Aided Verification (CAV'08)*, pages 517–529. Springer, 2008. doi:10.1007/978-3-540-70545-1\_49.
- 14 Jan Friso Groote, David N. Jansen, Jeroen J. A. Keiren, and Anton J. Wijs. An  $O(m \log n)$  algorithm for computing stuttering equivalence and branching bisimulation. *ACM Trans. Comput. Logic*, 18(2), jun 2017. doi:10.1145/3060140.
- 15 Orna Grumberg and David E. Long. Model checking and modular verification. In *Proceedings of the 2nd International Conference on Concurrency Theory, CONCUR '91*, volume 527 of *Lecture Notes in Computer Science*, pages 250–265. Springer, 1991. doi:10.1007/3-540-54430-5\_93.
- 16 Orna Grumberg and David E Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(3):843–871, 1994. doi:10.1145/177492.177725.
- 17 Bhargav S. Gulavani, Thomas A. Henzinger, Yamini Kannan, Aditya V. Nori, and Sriram K. Rajamani. SYNERGY: a new algorithm for property checking. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006*, pages 117–127. ACM, 2006. doi:10.1145/1181775.1181790.

- 18 Michel Habib, Christophe Paul, and Laurent Viennot. Partition refinement techniques: An interesting algorithmic tool kit. *Int. J. Found. Comput. Sci.*, 10(2):147–170, 1999. doi:[10.1142/S0129054199000125](https://doi.org/10.1142/S0129054199000125).
- 19 M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 453–462, October 1995. doi:[10.1109/SFCS.1995.492576](https://doi.org/10.1109/SFCS.1995.492576).
- 20 David N. Jansen, Jan Friso Groote, Jeroen J. A. Keiren, and Anton Wijs. An  $O(m \log n)$  algorithm for branching bisimilarity on labelled transition systems. In *Proceedings of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, pages 3–20. Springer, 2020. doi:[10.1007/978-3-030-45237-7\\_1](https://doi.org/10.1007/978-3-030-45237-7_1).
- 21 Antonín Kucera and Richard Mayr. Simulation preorder over simple process algebras. *Inf. Comput.*, 173(2):184–198, 2002. doi:[10.1006/inco.2001.3122](https://doi.org/10.1006/inco.2001.3122).
- 22 Antonín Kucera and Richard Mayr. Why is simulation harder than bisimulation? In *Proceedings of the 13th International Conference on Concurrency Theory, CONCUR 2002*, volume 2421 of *Lecture Notes in Computer Science*, pages 594–610. Springer, 2002. doi:[10.1007/3-540-45694-5\\_39](https://doi.org/10.1007/3-540-45694-5_39).
- 23 David Lee and Mihalis Yannakakis. Online Minimization of Transition Systems. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC '92*, pages 264–274, New York, NY, USA, 1992. ACM. doi:[10.1145/129712.129738](https://doi.org/10.1145/129712.129738).
- 24 Claire Loiseaux, Susanne Graf, Joseph Sifakis, Ahmed Bouajjani, and Saddek Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods Syst. Des.*, 6(1):11–44, 1995. doi:[10.1007/BF01384313](https://doi.org/10.1007/BF01384313).
- 25 Rupak Majumdar, Necmiye Ozay, and Anne-Kathrin Schmuck. On abstraction-based controller design with output feedback. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–11, Sydney New South Wales Australia, April 2020. ACM. doi:[10.1145/3365365.3382219](https://doi.org/10.1145/3365365.3382219).
- 26 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, N. J., 1967.
- 27 Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987. doi:[10.1137/0216062](https://doi.org/10.1137/0216062).
- 28 Corina S. Pasareanu, Radek Pelánek, and Willem Visser. Concrete model checking with abstract matching and refinement. In *Proc. 17th International Conference on Computer Aided Verification, CAV 2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2005. doi:[10.1007/11513988\\_7](https://doi.org/10.1007/11513988_7).
- 29 Francesco Ranzato. An efficient simulation algorithm on Kripke structures. *Acta informatica*, 51(2):107–125, 2014. doi:[10.1007/s00236-014-0195-9](https://doi.org/10.1007/s00236-014-0195-9).
- 30 Francesco Ranzato and Francesco Tapparo. A new efficient simulation equivalence algorithm. In *Proceedings of the 22nd IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 171–180. IEEE Computer Society, 2007. doi:[10.1109/LICS.2007.8](https://doi.org/10.1109/LICS.2007.8).
- 31 Francesco Ranzato and Francesco Tapparo. An efficient simulation algorithm based on abstract interpretation. *Information and Computation*, 208(1):1–22, January 2010. doi:[10.1016/j.ic.2009.06.002](https://doi.org/10.1016/j.ic.2009.06.002).
- 32 Michael Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, Boston, 2nd ed edition, 2006.
- 33 Li Tan and Rance Cleaveland. Simulation revisited. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, volume 2031 of *Lecture Notes in Computer Science*, pages 480–495. Springer, 2001. doi:[10.1007/3-540-45319-9\\_33](https://doi.org/10.1007/3-540-45319-9_33).

## A Auxiliary Algorithms

### Algorithm 4 Sim

---

**Input:** A finite transition system  $G = (\Sigma, I, \rightarrow)$  and an initial qo  $R \in \text{QO}(\Sigma)$

```

1 while  $\exists x, x' \in \Sigma. x \rightarrow x' \wedge R(x) \not\subseteq \text{pre}(R(x'))$  do
2    $R(x) := R(x) \cap \text{pre}(R(x'))$ ;
3 end
4 return  $R$ ;

```

---

### Algorithm 5 SymbSim

---

**Input:** A transition system  $G = (\Sigma, I, \rightarrow)$  and an initial partition/relation  $\langle P, \tau \rangle$

```

1 while  $\exists B, C \in P. B \cap \text{pre}(C) \neq \emptyset \wedge \cup \tau(B) \not\subseteq \text{pre}(\cup \tau(C))$  do
2    $S := \text{pre}(\cup \tau(C))$ ;
3    $P_{\text{prev}} := P$ ;  $C_{\text{prev}} := C$ ;
4    $P := \text{Split}(P, S)$ ;
5   forall  $D \in P$  do  $\tau(D) := \{E \in P \mid E \subseteq \cup \tau(P_{\text{prev}}(D))\}$ ;
6   forall  $D \in P$  such that  $D \cap \text{pre}(C_{\text{prev}}) \neq \emptyset$  do  $\tau(D) := \{E \in \tau(D) \mid E \subseteq S\}$ ;
7 end
8 return  $\langle P, \tau \rangle$ ;

```

---

## B Proofs

► **Theorem 4.1** (Correctness of Algorithm 1). *Let  $\langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \wp(\Sigma)$  be the output of Algorithm 1 on input  $R_i \in \text{QO}(\Sigma)$  where  $\Sigma$  is finite, and 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:*

$$\{R_{\text{sim}}(x) \mid x \in \Sigma, R_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset\} = \{R(x) \mid x \in \Sigma, R(x) \cap \sigma \neq \emptyset\}, \quad (4)$$

$$\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\} \subseteq \{B \in P \mid B \cap \sigma \neq \emptyset\}. \quad (5)$$

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

- (i) For any 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 any 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,  $x \in R(x)$  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  $I \rightarrow^n x' \rightarrow x$ , and by inductive hypothesis,  $R(x') \cap \sigma \neq \emptyset$ . Therefore, since  $V = \emptyset$ ,  $R(x') \subseteq \text{pre}(R(x))$  must hold. Thus,  $\text{pre}(R(x)) \cap \sigma \neq \emptyset$ , so that  $R(x) \cap \text{post}(\sigma) \neq \emptyset$ . 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 4. 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  $x \rightarrow y$ ,  $R'(x) = R(x) \not\subseteq \text{pre}(R'(y))$ . We have  $R(x) \cap \sigma \neq \emptyset$ ,  $x \rightarrow y$  and  $V = \emptyset$  entail  $R(x) \subseteq \text{pre}(R(y))$ . Hence,  $R(x) \cap \sigma \neq \emptyset$  implies  $R(y) \cap \text{post}(\sigma) \neq \emptyset$ , 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  $R(x) \not\subseteq \text{pre}(R(y))$ . Moreover,  $R(x) \cap \sigma \neq \emptyset$ ,  $x \rightarrow y$  and  $V = \emptyset$  imply  $R(x) \subseteq \text{pre}(R(y))$ , 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 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 (4).

( $\subseteq$ ) 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 (5). 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 (4),  $R_{\text{sim}}(y) = R(y)$  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 (4),  $R_{\text{sim}}(z) = R(z)$ . 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 (5).  $\blacktriangleleft$

► **Theorem 4.3** (Correctness of Algorithm 2). *Let  $\langle P, \tau, \sigma \rangle \in \text{Part}(\Sigma) \times \text{Rel}(P) \times \wp(\Sigma)$  be the output of Algorithm 2 on input  $R_i \in \text{QO}(\Sigma)$ , and 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.  $R_i$ , where  $P_{\text{sim}}$  is assumed to be a recursively enumerable set. Then:*

$$\{R_{\text{sim}}(x) \mid x \in \Sigma, R_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset\} = \{\cup \tau(P(x)) \mid x \in \Sigma, \cup \tau(P(x)) \cap \sigma \neq \emptyset\}, \quad (6)$$

$$\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\} \subseteq \{B \in P \mid \cup \tau(B) \cap \sigma \neq \emptyset\}. \quad (7)$$

**Proof.** Observe that the following invariant property is true for the repeat-until loop of Algorithm 2: for all  $x \in \Sigma$ ,

$$P(x) = \{y \in \Sigma \mid \cup \tau(P(x)) = \cup \tau(P(y))\}. \quad (10)$$

In fact, (10) holds at the beginning for the initializations at lines 1–2, and (10) is preserved by the refine block at lines 12–18.

The following facts on the output tuple  $\langle P, \tau, \sigma \rangle$  hold:

- (i) For any iteration,  $\tau$  is reflexive.

This holds because at the beginning  $\tau$  is a quasiorder and the update and refine code block of  $\tau$  at lines 16–18 of Algorithm 2 preserves the reflexivity of  $\tau$ , in particular the refinement statement at line 18.

- (ii) For all  $y \in \Sigma$ ,  $R_{\text{sim}}(y) \subseteq \cup\tau(P(y))$ .

This holds because the *Refine* block of Algorithm 2 is always correct, so that  $R_{\text{sim}}(y) \subseteq \cup\tau(P(y)) \subseteq R_i(y)$  holds for any iteration of Algorithm 2.

- (iii)  $x \in \text{post}^*(I) \Rightarrow \cup\tau(P(x)) \cap \sigma \neq \emptyset$ .

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  $\tau$  is reflexive,  $x \in \tau(P(x))$  and therefore  $\cup\tau(P(x)) \cap (I \cup \text{post}(\sigma)) \neq \emptyset$ . Hence,  $U = \emptyset$  implies that  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ . If  $n > 0$  then  $I \rightarrow^n x' \rightarrow x$ , so that  $P(x') \cap \text{pre}(P(x)) \neq \emptyset$ . By inductive hypothesis,  $\cup\tau(P(x')) \cap \sigma \neq \emptyset$ . Therefore, since  $V = \emptyset$ ,  $\cup\tau(P(x')) \subseteq \text{pre}(\cup\tau(P(x)))$  must hold. Thus,  $\text{pre}(\cup\tau(P(x))) \cap \sigma \neq \emptyset$ , so that  $\cup\tau(P(x)) \cap \text{post}(\sigma) \neq \emptyset$ . Hence,  $U = \emptyset$  implies  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ .

- (iv) For all  $x \in \Sigma$ ,  $\cup\tau(P(x)) \cap \sigma \neq \emptyset \Rightarrow \cup\tau(P(x)) = R_{\text{sim}}(x)$ .

Let  $\text{Sim}$  be the basic simulation algorithm,  $\text{SymbSim}$  be the basic symbolic simulation algorithm in [31] as recalled in Algorithm 5, and let  $R_{\langle P, \tau \rangle} \triangleq \{(x, y) \in \Sigma^2 \mid y \in \cup\tau(P(x))\}$ . By (ii),  $R_{\text{sim}} \subseteq R_{\langle P, \tau \rangle} \subseteq R_i$ . Thus,  $\text{Sim}(R_{\langle P, \tau \rangle}) = R_{\text{sim}}$  because  $R_{\text{sim}} = \text{Sim}(R_{\text{sim}}) \subseteq \text{Sim}(R_{\langle P, \tau \rangle}) \subseteq \text{Sim}(R_i) = R_{\text{sim}}$ . Assume, by contradiction, that  $\mathbb{S} \triangleq \{P(x) \in P \mid \cup\tau(P(x)) \cap \sigma \neq \emptyset, \cup\tau(P(x)) \neq R_{\text{sim}}(x)\} \neq \emptyset$ , so that for each block  $P(x) \in \mathbb{S}$ ,  $\tau(P(x))$  will be refined at some iteration of  $\text{SymbSim}(\langle P, \tau \rangle)$  at lines 5–6. Then, let  $P(x) \in \mathbb{S}$  be the first block in  $\mathbb{S}$  such that  $\tau(P(x))$  is refined by  $\text{SymbSim}$  at some iteration  $it \in \mathbb{N}$  (this will happen because  $P_{\text{sim}}$  is assumed to be recursively enumerable so that  $\text{SymbSim}(\langle P, \tau \rangle)$  will eventually refine all the blocks in  $\mathbb{S}$ ) whose current partition/relation pair is  $\langle P', \tau' \rangle$ . Thus, each block  $P(z) \in \mathbb{S}$  is such that  $P(z) = P'(z)$  and  $\tau(P(z)) = \tau'(P'(z))$  because no block in  $\mathbb{S}$  was refined by  $\text{SymbSim}$  before this iteration  $it$ . Let  $P'(y) \subseteq P(y)$  be the block of  $P'$  used by  $\text{SymbSim}$  in this iteration  $it$  to refine  $\tau(P(x))$ , so that  $P(x) \cap \text{pre}(P'(y)) \neq \emptyset$ ,  $\cup\tau'(P'(y)) = \cup\tau(P(x)) \not\subseteq \text{pre}(\cup\tau'(P'(y)))$ .

We have that  $P(x) \cap \text{pre}(P'(y)) \neq \emptyset$  and  $P'(y) \subseteq P(y)$  imply  $P(x) \cap \text{pre}(P(y)) \neq \emptyset$ . Thus,  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ ,  $P(x) \cap \text{pre}(P(y)) \neq \emptyset$  and  $V = \emptyset$  entail  $\cup\tau(P(x)) \subseteq \text{pre}(\cup\tau(P(y)))$ . Hence,  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$  implies  $\cup\tau(P(y)) \cap \text{post}(\sigma) \neq \emptyset$ , so that  $U = \emptyset$  entails  $\cup\tau(P(y)) \cap \sigma \neq \emptyset$ . If  $P(y) \notin \mathbb{S}$  then  $\cup\tau(P(y)) \cap \sigma \neq \emptyset$  implies  $\cup\tau(P(y)) = R_{\text{sim}}(y) \subseteq \cup\tau'(P'(y)) \subseteq \cup\tau(P(y))$ , so that  $\cup\tau'(P'(y)) = \cup\tau(P(y))$  holds. If  $P(y) \in \mathbb{S}$  then, since  $P(x)$  is the first block such that  $\tau(P(x))$  is refined by  $\text{SymbSim}$ , we have that  $\cup\tau'(P'(y)) = \cup\tau(P(y))$  holds. Thus, in both cases,  $\cup\tau'(P'(y)) = \cup\tau(P(y))$  must hold, so that  $\cup\tau(P(x)) \not\subseteq \text{pre}(\cup\tau(P(y)))$ . Moreover,  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ ,  $P(x) \cap \text{pre}(P(y)) \neq \emptyset$  and  $V = \emptyset$  imply  $\cup\tau(P(x)) \subseteq \text{pre}(\cup\tau(P(y)))$ , which is therefore a contradiction. Thus,  $\mathbb{S} = \emptyset$ , and, in turn,  $\cup\tau(P(x)) = R_{\text{sim}}(x)$ .

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

This follows because  $\sigma$  is updated at line 11 of Algorithm 2 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 (6).

( $\subseteq$ ) 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),  $\cup\tau(P(s)) \cap \sigma \neq \emptyset$ . By (iv),  $\cup\tau(P(s)) = R_{\text{sim}}(s)$ . 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),  $R_{\text{sim}}(x) \subseteq \cup\tau(P(x))$ , we obtain  $R_{\text{sim}}(s) \subseteq \cup\tau(P(x))$ . Thus,  $\cup\tau(P(s)) \subseteq \cup\tau(P(x))$  holds, so that  $\cup\tau(P(s)) \cap \sigma \neq \emptyset$  entails  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ , and in turn, by (iii),  $\cup\tau(P(x)) = R_{\text{sim}}(x)$ .

( $\supseteq$ ) Let  $x \in \Sigma$  such that  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ . By (iv),  $\cup\tau(P(x)) = R_{\text{sim}}(x)$ , 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 (7). Let  $x \in \Sigma$  such that  $P_{\text{sim}}(x) \cap \text{post}^*(I) \neq \emptyset$ . By (10), we have that for all  $x \in \Sigma$ ,  $P(x) = \{y \in \Sigma \mid \cup\tau(P(x)) = \cup\tau(P(y))\}$ . 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 (6) of Theorem 4.3,  $R_{\text{sim}}(y) = \cup\tau(P(y))$  and  $\cup\tau(P(y)) \cap \sigma \neq \emptyset$ . In particular,  $\cup\tau(P(x)) = R_{\text{sim}}(x) = R_{\text{sim}}(y) = \cup\tau(P(y))$ . Therefore,  $P_{\text{sim}}(x) \subseteq \{y \in \Sigma \mid \cup\tau(P(x)) = \cup\tau(P(y))\} = P(x)$ . On the other hand, if  $z \in P(x)$  then  $\cup\tau(P(z)) = \cup\tau(P(x))$ , so that  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$  implies  $\cup\tau(P(z)) \cap \sigma \neq \emptyset$ . Thus, by (6) of Theorem 4.3,  $R_{\text{sim}}(z) = \cup\tau(P(z))$ . Hence,  $R_{\text{sim}}(z) = \cup\tau(P(z)) = \cup\tau(P(x)) = R_{\text{sim}}(x)$ , thus proving that  $P(x) \subseteq P_{\text{sim}}(x)$ , and therefore  $P(x) = P_{\text{sim}}(x)$  holds. Moreover,  $\cup\tau(P(x)) \cap \sigma \neq \emptyset$ , thus proving (7).  $\blacktriangleleft$

► **Theorem 5.1** (Correctness of Algorithm 3). *Let  $\langle P, \tau, \sigma \rangle \in \text{Part}(\Sigma) \times \text{Rel}(\wp(\Sigma)) \times \wp(\Sigma)$  be the output of Algorithm 3 on input  $R_i \in \text{QO}(\Sigma)$ , and  $R_{\text{sim}} \in \text{QO}(\Sigma)$  and  $P_{\text{sim}} \in \text{Part}(\Sigma)$  be, resp., the simulation  $qo$  and partition w.r.t.  $R_i$ , where  $P_{\text{sim}}$  is assumed to be a recursively enumerable set. Then:*

$$\{R_{\text{sim}}(x) \mid x \in \text{post}^*(I)\} = \{\cup\tau(P(x)) \mid x \in \sigma\}, \quad (8)$$

$$\{B \in P_{\text{sim}} \mid B \cap \text{post}^*(I) \neq \emptyset\} = \{B \in P \mid B \cap \sigma \neq \emptyset\}. \quad (9)$$

**Proof.** The following facts on the output triple  $\langle P, \tau, \sigma \rangle$  of Algorithm 3 hold:

(i) For all  $x \in \Sigma$ ,  $P_{\text{sim}}(x) \subseteq P(x)$  and  $R_{\text{sim}}(x) \subseteq \cup\tau(P(x))$ .

This holds because the *Refine* block of Algorithm 3 is always correct, so that, for all  $x \in \Sigma$ ,  $P_{\text{sim}}(x) \subseteq P(x) \subseteq P_{R_i}(x)$  and  $R_{\text{sim}}(x) \subseteq \cup\tau(P(x)) \subseteq R_i(x)$  holds for any iteration of Algorithm 3.

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

This is proven by induction on  $n \in \mathbb{N}$  such that  $x \in \text{post}^n(I)$ . For  $n = 0$ ,  $x \in I$ , so that  $P(x) \cap I \neq \emptyset$ , and, in turn,  $\cup\tau(P(x)) \cap I \neq \emptyset$ , therefore  $U = \emptyset$  implies  $P(x) \cap \sigma \neq \emptyset$ . 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,  $P(y) \cap \sigma \neq \emptyset$ . Since  $y \rightarrow x$ ,  $P(y) \cap \text{pre}(P(x)) \neq \emptyset$ , so that  $P(y) \cap \sigma \neq \emptyset$  and  $V = \emptyset$  imply  $\cup\tau(P(y)) \subseteq \text{pre}(\cup\tau(P(x)))$ . Thus, since  $\tau$  is reflexive,  $P(y) \subseteq \cup\tau(P(y)) \subseteq \text{pre}(\cup\tau(P(x)))$  holds, that together with  $P(y) \cap \sigma \neq \emptyset$  implies  $\text{pre}(\cup\tau(P(x))) \cap \sigma \neq \emptyset$ , i.e.,  $\cup\tau(P(x)) \cap \text{post}(\sigma) \neq \emptyset$ . Hence,  $U = \emptyset$  implies  $P(x) \cap \sigma \neq \emptyset$ .

(iii)  $P(x) \cap \sigma \neq \emptyset \Rightarrow P(x) = P_{\text{sim}}(x)$ .

Let *SymbSim* be the symbolic simulation algorithm [31] taking in input a partition/relation pair  $\langle P, \tau \rangle$  and computing, for finite state systems, as output  $\langle P_{\text{sim}}, \tau_{\text{sim}} \rangle$  with  $R_{\text{sim}} = \lambda x. \cup\tau_{\text{sim}}(P_{\text{sim}}(x))$ . By (i),  $P_{\text{sim}} \preceq P \preceq P_{R_i}$  and  $R_{\text{sim}}(z) \subseteq \cup\tau(P(z)) \subseteq R_i(z)$ , for all  $z \in \Sigma$ . Moreover,  $\text{SymbSim}(\langle P, \tau \rangle) = P_{\text{sim}}$  because  $\langle P_{\text{sim}}, \tau_{\text{sim}} \rangle = \text{SymbSim}(\langle P_{\text{sim}}, \tau_{\text{sim}} \rangle) \subseteq \text{SymbSim}(\langle P, \tau \rangle) \subseteq \text{SymbSim}(\langle P_{R_i}, R_i \rangle) = \langle P_{\text{sim}}, \tau_{\text{sim}} \rangle$ . Let us assume, by contradiction, that  $\mathbb{S} \triangleq \{P(x) \in P \mid P(x) \neq P_{\text{sim}}(x), P(x) \cap \sigma \neq \emptyset\} \neq \emptyset$ , so that every block in  $\mathbb{S}$  will be split at some iteration of *SymbSim* on input  $\langle P, \tau \rangle$ . Let  $P(x) \in \mathbb{S}$  be the first block in  $\mathbb{S}$  to be split by *SymbSim* at some iteration  $it$  whose current partition/relation is  $\langle P', \tau' \rangle$  with  $P' \preceq P$  and  $\tau' \subseteq \tau$ , so that each block in  $\mathbb{S}$  is a block for  $P'$  since no block in  $\mathbb{S}$  was split before this iteration  $it$ . Let  $P'(y) \subseteq P(y)$  be the block of  $P'$  used in this iteration  $it$  to split  $P(x)$ , so that  $P(x) \cap \text{pre}(P'(y)) \neq \emptyset$ ,  $\cup\tau'(P(x)) \not\subseteq \text{pre}(\cup\tau'(P'(y)))$  and  $P(x) \not\subseteq \text{pre}(\cup\tau'(P'(y)))$ , namely,  $P(x)$  is actually split by  $S = \text{pre}(\cup\tau'(P'(y)))$ . We have that  $P(x) \cap \text{pre}(P'(y)) \neq \emptyset$  implies



$P(x) \cap \text{pre}(P(y)) \neq \emptyset$ , so that  $P(x) \cap \sigma \neq \emptyset$  and  $V = \emptyset$  entail  $\cup\tau(P(x)) \subseteq \text{pre}(\cup\tau(P(y)))$ . Hence, by reflexivity,  $P(x) \subseteq \cup\tau(P(x))$ , so that

$$P(x) \subseteq \cup\tau(P(x)) \subseteq \text{pre}(\cup\tau(P(y))) \quad (11)$$

and, in turn,  $\text{post}(\sigma) \cap \cup\tau(P(y)) \neq \emptyset$ . Therefore,  $U = \emptyset$  implies

$$P(y) \cap \sigma \neq \emptyset. \quad (12)$$

Let us also show that:

$$\forall u \in \Sigma. P(u) \cap \sigma \neq \emptyset \Rightarrow P(u) = P'(u). \quad (13)$$

In fact, if  $P(u) \in \mathbb{S}$ , then  $P(u)$  was not split before *it* since  $P(x)$  is the first block of  $\mathbb{S}$  to be split, so that  $P(u) = P'(u)$ ; on the other hand, if  $P(u) \notin \mathbb{S}$  then  $P(u) \cap \sigma \neq \emptyset$  implies  $P(u) = P_{\text{sim}}(u)$  and since  $P_{\text{sim}} \preceq P' \preceq P$  then  $P(u) = P'(u)$ .

We now show that, for any  $P'$  satisfying (13):

$$\cup\tau'(P'(y)) = \cup\tau(P(y)). \quad (14)$$

Let us assume, by contradiction, that  $\cup\tau'(P'(y)) \neq \cup\tau(P(y))$ , namely  $\cup\tau'(P'(y)) \subsetneq \cup\tau(P(y))$ .

By (12),  $P(y) \cap \sigma \neq \emptyset$ , hence, by (13),  $P(y) = P'(y)$ , so that  $\cup\tau(P(y)) = \cup\tau(P'(y))$  holds, and, in turn  $\cup\tau'(P'(y)) = \cup\tau'(P(y)) \subsetneq \cup\tau(P(y))$ . This means that at some iteration of  $\text{SymbSim}(\langle P, \tau \rangle)$  preceding *it*, the relation  $\tau(P(y))$  has been refined to  $\tau'(P(y))$  without splitting the block  $P(y)$ . Let  $it_1 < it$  be the first iteration of  $\text{SymbSim}(\langle P, \tau \rangle)$  where  $\tau(P(y))$  is refined without splitting  $P(y)$ , and let  $\langle P'', \tau'' \rangle$  be the partition/relation pair at this iteration  $it_1$ , so that  $\langle P', \tau' \rangle \subseteq \langle P'', \tau'' \rangle \subseteq \langle P, \tau \rangle$  holds. Let  $P''(z)$  be the block of  $P''$  used at this iteration  $it_1$  for refining  $\tau(P(y))$  without splitting the block  $P(y)$ , so that  $P(y) \cap \text{pre}(P''(z)) \neq \emptyset$ ,  $\cup\tau(P(y)) = \cup\tau''(P(y)) \not\subseteq \text{pre}(\cup\tau''(P''(z)))$  and  $P(y) \subseteq \text{pre}(\cup\tau''(P''(z)))$ . By reflexivity,  $P''(z) \subseteq \cup\tau''(P''(z))$ , so that  $\text{pre}(P''(z)) \subseteq \text{pre}(\cup\tau''(P''(z)))$ . Thus,  $P(y) \cap \text{pre}(\cup\tau''(P''(z))) \neq \emptyset$ . Moreover,  $P(y) \subseteq \text{pre}(\cup\tau''(P''(z))) \subseteq \text{pre}(\cup\tau(P(z)))$ . Hence, since, by (12),  $P(y) \cap \sigma \neq \emptyset$ , it turns out that  $\cup\tau(P(z)) \cap \text{post}(\sigma) \neq \emptyset$ , so that  $U = \emptyset$  entails  $P(z) \cap \sigma \neq \emptyset$ . Thus, by (13),  $P(z) \cap \sigma \neq \emptyset$  implies  $P(z) = P'(z)$ , that together with  $P' \preceq P'' \preceq P$ , entails  $P(z) = P''(z)$ , so that  $P(y) \cap \text{pre}(P(z)) \neq \emptyset$  holds. It turns out that  $\cup\tau''(P(z)) \neq \cup\tau(P(z))$ : in fact, if we assume, by contradiction, that  $\cup\tau''(P(z)) = \cup\tau(P(z))$  then we would have  $\cup\tau(P(y)) \not\subseteq \text{pre}(\cup\tau''(P''(z))) = \text{pre}(\cup\tau(P(z)))$ , meaning that  $(P(y), P(z)) \in V$ , which would be a contradiction to  $V = \emptyset$ . Thus,  $P''(z) = P(z)$  and  $\cup\tau''(P(z)) \subsetneq \cup\tau(P(z))$  means that we can find the first iteration  $it_2$  of  $\text{SymbSim}(\langle P, \tau \rangle)$  preceding  $it_1$ , i.e.  $it_2 < it_1$ , where the relation  $\tau(P(z))$  has been refined to  $\tau''(P(z))$  without splitting the block  $P(z)$ . This therefore leads to the contradiction of constructing an infinite sequence of iterations  $it > it_1 > it_2 > \dots$ , whereas the iterations preceding *it* are finitely many. We can therefore conclude that  $\cup\tau'(P'(y)) \neq \cup\tau(P(y))$  cannot hold, namely, we proved (14). Therefore, by (11),  $P(x) \subseteq \text{pre}(\cup\tau(P(y))) = \text{pre}(\cup\tau'(P'(y)))$ , which is a contradiction to  $P(x) \not\subseteq \text{pre}(\cup\tau'(P'(y)))$ .

(iv)  $P(x) \cap \sigma \neq \emptyset \Rightarrow R_{\text{sim}}(x) = \cup\tau(P(x))$ .

Let  $\langle P_{\text{sim}}, \tau_{\text{sim}} \rangle$  be the output of  $\text{SymbSim}$  on input  $\langle P, \tau \rangle$ , then by correctness of  $\text{SymbSim}$ ,  $R_{\text{sim}}(x) = \cup\tau_{\text{sim}}(P_{\text{sim}}(x))$ . Moreover, by (iii), we get that no block in  $P^\sigma$  was split in  $P_{\text{sim}}$  and thus by (14) we conclude  $\cup\tau_{\text{sim}}(P_{\text{sim}}(x)) = \cup\tau(P(x))$ , and thus  $R_{\text{sim}}(x) = \cup\tau(P(x))$ .

Let us now show (9):  $\{P_{\text{sim}}(x) \mid x \in \text{post}^*(I)\} = \{P(x) \mid x \in \sigma\}$ . On the one hand, if  $P_{\text{sim}}(x) \in P_{\text{sim}}$  for some  $x \in \text{post}^*(I)$  then, by (ii),  $P(x) \cap \sigma \neq \emptyset$ , so that  $P(x) = P(s)$  for some  $s \in \sigma$ , and, by (iii),  $P(s) = P(x) = P_{\text{sim}}(x)$ . On the other hand, if  $P(s) \in P$  for some  $s \in \sigma$  then, by (iii),  $P(s) = P_{\text{sim}}(s)$ , and, since  $\sigma \subseteq \text{post}^*(I)$ ,  $s \in \text{post}^*(I)$ .

Finally, we prove (8):  $\{R_{\text{sim}}(x) \mid x \in \text{post}^*(I)\} = \{\cup \tau(P(x)) \mid x \in \sigma\}$ . On the one hand, if  $x \in \text{post}^*(I)$  then, by (ii),  $P(x) \cap \sigma \neq \emptyset$ , so that,  $P(x) = P(s)$  for some  $s \in \sigma$ , and, in turn by (iv),  $R_{\text{sim}}(x) = \cup \tau P(x) = \cup \tau P(s)$ . On the other hand, if  $s \in \sigma$  then, by (iv),  $R_{\text{sim}}(s) = \cup \tau(P(s))$ , and, since  $\sigma \subseteq \text{post}^*(I)$ ,  $s \in \text{post}^*(I)$ .  $\blacktriangleleft$