Online Simulation Reduction

Pierre Ganty\textsuperscript{1}, Nicolas Manini\textsuperscript{1,2}, and Francesco Ranzato\textsuperscript{3}

\textsuperscript{1} IMDEA Software Institute, Spain
\textsuperscript{2} Universidad Politécnica de Madrid, Spain
\textsuperscript{3} University of Padova, Italy

\{pierre.ganty, nicolas.manini\}@imdea.org
francesco.ranzato@unipd.it

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 \textit{online simulation reduction} to reflect some significant dissimilarities with respect to Lee and Yannakakis’ online bisimulation minimization. Indeed, by means of a reduction from 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 an 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 algorithm which is complete for simulation equivalence but, due to the undecidability result, reveals unavoidable limitations on its termination. Finally, we report on a prototype implementation.

Keywords: Simulation · Reachability · Online Algorithm · System Reduction.

1 Introduction

One way to tackle the well-known state explosion problem arising in formal verification \cite{Henzinger1996} is to minimize the transition system to analyze, for instance, by computing the quotient of the system with respect to some behavioral equivalence relation between states, typically bisimulation or simulation equivalence \cite[Chapter 13]{Henzinger1996}. To further reduce the size of the quotiented 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 \textit{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 \cite{Lee1992,Yannakakis1992}. 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.

Contributions. Our starting point was the well-known online algorithm by Lee and Yannakakis [24], here referred to as the LY algorithm, which relies on an intricate interleaving of reachability and bisimulation computation and comes with several complexity guarantees. Roughly speaking, the runtime of LY 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 this is the case, 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, yet the similarity quotient retains enough precision for checking all linear temporal formulas or branching temporal logic formulas without quantifier switches [26, 17, 18, 11]. Moreover, infinite state systems like 2D rectangular automata may have infinite bisimilarity quotients, nonetheless finite similarity quotients [21].

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 system, 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 [8] for their non-online simulation minimization algorithm. While in the bisimulation case transitions between reachable blocks defined using a ∃∃ policy (i.e., $B \rightarrow B'$ iff $\exists s \in B. \exists s' \in B', s \rightarrow s'$) or a ∀∃ 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 ∀∃ transitions [8].

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 natural counterpart to the notion of “equivalence class” is 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 conceptually so 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 [20]). On the other hand, the algorithms com-
puting the simulation preorder, such as the well-known Henzinger, Henzinger and Kopke procedure [21], 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 a first algorithm which performs online reduction in a sound and complete way for the simulation preorder while yielding a sound over-approximation for simulation equivalence. In Section 5 we investigate a second algorithm for which we prove some completeness result with respect to simulation equivalence. Both algorithms terminate for finite state systems. Regarding their correctness, our results go beyond finite state systems and cover some infinite state systems as well. Finally, in Section 6 we report on a prototype implementation of our algorithm.

Appendix B contains the correctness and termination proofs for finite state systems. Appendix C discusses the infinite state case.

Related Work. There is a large body of work [12,21,31,32,34,33,9,16,15,36,4] 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 ∀CTL* formulas [17,18]. Kučera and Mayr [22,23] 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 [24] is the state-of-the-art, and, prior to it, Bouajjani et al. [56] gave the first solutions. These algorithms have been revisited by Alur and Henzinger in a chapter of their unpublished book on computer-aided verification [2, Chapter 4], as well as by the theoretical and experimental comparison made by Fisler and Vardi [14]. 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 [19,29] and hybrid systems verification [27].

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) ∈ R} is the converse relation of R. Moreover, for a given set S ∈ \(\wp(\Sigma)\) we denote \(R^S = \{R(x) \mid x ∈ \Sigma, R(x) ∩ S ≠ \emptyset\}\). A relation R ∈ Rel(\(\Sigma\)) is a quasiorder (qo, sometimes also called preorder) if it is reflexive and transitive, and \(\text{QO}(\Sigma) \triangleq \{R ∈ \text{Rel}(\Sigma) \mid R \text{ is a quasiorder}\}\) denotes the set of quasiorders on \(\Sigma\). Analogously \(R ∈ \text{Rel}(\Sigma)\) is an equivalence if it is a symmetric qo. Equivalences induce partitions where a partition of \(\Sigma\) consists of pairwise disjoint nonempty subsets of \(\Sigma\), called blocks, whose union is \(\Sigma\). An equivalence relation in \(\text{Rel}(\Sigma)\) induces a partition of \(\Sigma\) where each block is an equivalence class, and vice versa. Define Part(\(\Sigma\)) to be the set of partitions of \(\Sigma\). Given a partition \(P ∈ \text{Part}(\Sigma)\), \(P(s), P(S)\) and \(P^S\) (for \(S ∈ \varphi(\Sigma), s ∈ \Sigma\)) are well defined thanks to the equivalence underlying \(P\). In particular, \(P(s)\) is the block including \(s\), \(P(S) = \bigcup\{P(s) ∈ P \mid s ∈ S\}\), and \(P^S = \{P(s) ∈ P \mid s ∈ S\} ∈ \text{Part}(P(S))\).

**Transition systems, Simulation and Bisimulation.** Let \(G = (\Sigma, I, L, →)\) be a (labeled) transition system, where \(\Sigma\) is a (possibly infinite) set of states, \(I ⊆ \Sigma\) is a subset of initial states, \(L\) is a finite set of action labels, and \(→ ∈ \varphi(\Sigma × L × \Sigma)\) is the transition relation, where we denote \((x, a, y) ∈ →\) as \(x → \stackrel{a}{→} y\) and moreover \(x → y \triangleq ∃ a ∈ L, x \stackrel{a}{→} y\). When the set \(L\) is a singleton, we leverage the previous notation when writing \(x → y\), and equivalently we also say that \((x, y) ∈ →\) when no ambiguity arises. Given \(a ∈ L\) define \(\text{post}_a : \varphi(\Sigma) → \varphi(\Sigma)\) as the usual successor transformer \(\text{post}_a(X) \triangleq \{y ∈ \Sigma \mid \exists x ∈ X, x → \stackrel{a}{→} y\}\), and, dually, \(\text{pre}_a : \varphi(\Sigma) → \varphi(\Sigma)\) is the predecessor \(\text{pre}_a(X) \triangleq \{y ∈ \Sigma \mid \exists x ∈ X, y → \stackrel{a}{→} x\}\). Moreover, we define \(\text{post} : \varphi(\Sigma) → \varphi(\Sigma)\) as \(\text{post}(X) \triangleq \bigcup_{a ∈ L} \text{post}_a(X)\) and, symmetrically, \(\text{pre} : \varphi(\Sigma) → \varphi(\Sigma)\) as \(\text{pre}(X) \triangleq \bigcup_{a ∈ L} \text{pre}_a(X)\). Thus, \(\text{post}^*(I) = \bigcup_{n ∈ \mathbb{N}} \text{post}^n(I)\) is the set of reachable states (from the initial states) of \(G\).

Given an (initial) qo \(R_i ∈ \text{QO}(\Sigma)\) (e.g., \(R_i\) can be the partition induced by the initial states or by some labeling of a Kripke structure), a relation \(R ∈ \text{Rel}(\Sigma)\) is a simulation on \(G\) w.r.t. \(R_i\) if: (1) \(R ⊆ R_i\); (2) \((s, t) ∈ R\) and \(s → s’\) imply \(∃ t’, t → \stackrel{a}{→} t’\) and \((s’, t’) ∈ R\). Given two principals \(R(s), R(s’)\) such that \(s → s’, R(s)\) is a-stable (or simply stable) w.r.t. \(R(s’)\) when \(R(s) ⊆ \text{pre}_a(R(s’))\), otherwise \(R(s)\) is called a-unstable (resp. stable) w.r.t. \(R(s’)\), and, in this case, \(R(s’)\) can refine \(R(s)\). The greatest (w.r.t. \(⊆\)) simulation relation on \(G\) exists and turns out to be a qo called simulation quasiorder of \(G\) w.r.t. \(R_i\), denoted by \(R_{\text{sim}} ∈ \text{QO}(\Sigma)\), while \(P_{\text{sim}} ∈ \text{Part}(\Sigma)\) is the simulation partition induced by the similarity equivalence \(R_{\text{sim}} ∩ (R_{\text{sim}})^{-1}\). A relation \(R ∈ \text{Rel}(\Sigma)\) is a bisimulation on \(G\) w.r.t. an (initial) partition \(P_t ∈ \text{Part}(\Sigma)\) if both \(R\) and \(R^{-1}\) are simulations on \(G\) w.r.t. \(P_t\). The greatest (w.r.t. \(⊆\)) bisimulation relation on \(G\) w.r.t. \(P_t\) exists and turns out to be an equivalence called bisimulation equivalence (or bisimilarity), whose corresponding bisimulation partition is denoted by \(P_{\text{bis}} ∈ \text{Part}(\Sigma)\).

## 3 Online Reduction for Simulation

The problem we address in this paper is the simultaneous computation of a reachability analysis and a simulation reduction of transition systems, that we call online simulation reduction, namely:
Given: An effectively presented transition system $G = (\Sigma, I, L, \rightarrow)$ and an initial $q_0 R_i \in QO(\Sigma)$.

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 (bi)simulation equivalence—is naturally defined as follows:

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

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, reachability for principals is not uniquely formulated, and as such, the two following different definitions of reachable principal of a reflexive relation $R \in \text{Rel}(\Sigma)$ can both be considered adequate, where, clearly, (3) $\subseteq$ (2). Example 3.1 shows that the inclusion may be strict:

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

(2)

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

(3)

**Example 3.1.** Consider the transition system $(\Sigma = \{0, 1\}, I = \{1\}, L = \{a\}, \rightarrow = \{(1, 1)\})$ and the initial $q_0 R_i = \Sigma \times \Sigma$. The simulation quasiorder induced by $R_i$ 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 $R_{\text{sim}}(1)$.

Let us remark 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 is, 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 [24].

**Theorem 3.1** and the 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, $H \subseteq Q$ is a set of halting states disjoint from $q_0$, and $\Delta \subseteq Q \times Q$ is a set of transitions of three types:

4 There is a finite encoding on the tape of the underlying Turing machine.
1. $q \xrightarrow{c\leftarrow c+1} q'$, which increases the counter $c$;
2. $q \xrightarrow{c\leftarrow c-1} q'$, which decreases the counter $c$; these transitions can only be taken if the counter $c$ stores a positive value;
3. $q \xrightarrow{c\leftarrow 0} q'$, 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 $(q, j_1, \ldots, j_{|C|}) \in Q \times \mathbb{N}^{|C|}$, where $q \in Q$ is a state and $j_1, \ldots, j_{|C|} \in \mathbb{N}$ are the values stored by the $|C|$ counters of $M$. A configuration $(q, j_1, \ldots, j_m)$ 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 $(j_1, \ldots, j_{|C|})$ if there exist configurations $c_1, \ldots, c_n$ such that: $c_1 \rightarrow \cdots \rightarrow c_n$; $c_1 = (q_0, j_1, \ldots, j_{|C|})$; and $c_n$ is halting.

The following halting problem for 2-CMs is known to be undecidable [28]:

Given: A 2-CM $M$.

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

We will show by reduction that since the halting problem for 2-CMs is undecidable then so must be the following reachability problem for simulation equivalence:

Given: A 2-CM $M = (Q, C, \Delta, q_0, H)$ and an effectively presented finite simulation partition $P_{\text{sim}}$ of the transition system $(\Sigma = Q \times \mathbb{N}^2, I = \{(q_0, 0, 0)\}, L = \{\{a\}, \rightarrow\})$ generated by $M$ w.r.t. an initial $q_0$ 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 $q_n$ for each non-halting state $q$ of $M$. Besides the transitions of $M$, $M_1$ has two additional transitions for each new state $q_n$ added at the previous step: $q \xrightarrow{c\leftarrow c+1} q_n$ and $q_n \xrightarrow{c\leftarrow c-1} q$, 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 $q_0$ 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
whether there is a path of arbitrary length in the transition system reaching \( B \),

decide whether \( v \) involved than in the bisimulation case. As the following example suggests, to

\( P \)

state per block of

for \( B \)

have that

\( \forall \)

\( \sim \)

post

\( \ast \)

Finally, it turns out that \( \{q_0, 0, 0\} \) can reach a halting configuration in \( M \) iff \( \post^*(\{q_0, 0, 0\}) \cap (H \times \mathbb{N}^2) \neq \emptyset \) \( \ast \)

\( \forall B \in P_{\sim}: \post^*(\{q_0, 0, 0\}) \cap B \neq \emptyset \). We have therefore shown the following result.

**Theorem 3.2.** The reachability problem for simulation equivalence 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 \( P_{\sim} \) of \( G \) w.r.t. \( R_i \), namely, the subset of blocks \( \{B \in P_{\sim} \mid B \cap \post^*(I) \neq \emptyset\} \), even under the hypothesis that \( P_{\sim} \) 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 \( P_{\sim} \) and \( P_{\bis} \) over finite transition systems. These two problems are obviously both decidable, since we can simply compute \( \post^*(I) \), and, then, by checking that \( \post^*(I) \cap B = \emptyset \) holds for every block \( B \) in \( P_{\bis} \) or \( P_{\sim} \).

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

For the simulation case, deciding the reachability of \( B \in P_{\sim} \) is more involved than in the bisimulation case. As the following example suggests, to decide whether \( B \in P_{\sim} \) 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 $S_1$ where $I = \{1\}$ and $L$ is a singleton.

\[
\begin{array}{ccccccccc}
1 & \rightarrow & 2 & \rightarrow & 3 & \cdots & \cdots & \cdots & \rightarrow & n & \rightarrow & 0
\end{array}
\]

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 $n \rightarrow 0$, yielding the following transition system $S_2$.

\[
\begin{array}{ccccccccc}
1 & \rightarrow & 2 & \rightarrow & 3 & \cdots & \cdots & \cdots & \rightarrow & n & \rightarrow & 0
\end{array}
\]

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 $S_1$ the state $0$ is reachable, while in $S_2$ it is not.

Let us observe that this is not the case for bisimulation, because we have that $P_{\text{bis}} = \{[0, 0], [1, 1], [2, 2], \ldots, [n, n]\}$ for $S_1$, while $P_{\text{bis}} = P_{\text{sim}}$ for $S_2$. ♦

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 $G = (\Sigma, I, L, \rightarrow)$ and the simulation partition $P_{\text{sim}}$ of $G$ w.r.t. an initial $q_0$ relation.

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

We show the NL-hardness result by means of a reduction using the st-connectivity problem for leveled directed graphs. A *leveled directed graph* is a directed graph $G$ whose nodes are partitioned in $k > 0$ levels $A_1, A_2, \ldots, A_k$, 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 [35, p. 333].

Now, we 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 $N = A_1 \cup A_2 \cup \cdots \cup A_k$, edges $E$, and two nodes $s \in A_1$, $t \in A_k$, define the transition system $T_s$ by taking $G$ with set of initial states $I = \{s\}$.
Next, consider as initial qo relation \( R_i = \{ \{ t \} \times N \} \cup \{ (N \smallsetminus \{ t \}) \times (N \smallsetminus \{ t \}) \} \).

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 \( R_i(N \smallsetminus \{ t \}) \subseteq \text{pre}(R_i(N \smallsetminus \{ t \})) \), \( R_i(N \smallsetminus \{ t \}) \subseteq \text{pre}(R_i(\{ t \})) \), 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 \( P_{\text{sim}} = \{ N \smallsetminus \{ t \}, \{ t \} \} \). 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, \( (N \smallsetminus \{ t \}) \cap \text{post}^*(I) \neq \emptyset \) since \( I = \{ s \} \) and \( s \in N \smallsetminus \{ t \} \); 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 \((n, m)\) 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 it is in L.

4 A Sound Online Reduction Algorithm

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

Algorithm 4 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 \( I \), so that \( R^* \triangleq \{ R(x) \mid R(x) \cap \sigma \neq \emptyset \} \) 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 \( R^* \) and the set \( V \) of unstable pairs of principals. A principal \( R(x) \) belongs to \( U \) if it contains an initial state or a successor of a state already known to be reachable. A triple \((a, x, x')\) 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 \( R(x) \not\subseteq \text{pre}_a(R(x')) \). The algorithm either updates the reachability information for some principal in \( U \) or stabilizes the pair of principals associated to \((a, x, x') \in V \) by refining \( R(x) \). In our pseudocode we use a nif statement which is a nondeterministic choice between guarded commands. In that statement we have three guarded commands: either the Search (lines 7–10)
Algorithm 1: Sound Algorithm

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

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

or the \text{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 \text{Search} and \text{Refine}, possibly followed by the return at line 14. Observe that the guards are such that when the algorithm terminates neither \text{Search} nor \text{Refine} are enabled.

A principal \( R(x) \) is refined at line 13 only if it is known to be currently reachable. Upon termination, \( R^* \) is the set of reachable principals of \( R_{\text{sim}} \) (1.a) below. However, \( R \) may well contain unstable principals, so that, in general, \( R \) and \( R_{\text{sim}} \) do not coincide. Turning to the simulation partition, we find that Algorithm 1 yields a sound overapproximation of \( P_{\text{sim}}^{\text{post}}(I) \) (1.b) below.

Theorem 4.1 (Correctness of Algorithm 1). Let \( \langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \varphi(\Sigma) \) be the output of Algorithm 1 on input \( G \) with \( |\Sigma| \in \mathbb{N} \), \( R_i \in \text{QO}(\Sigma) \) and \( \sigma_i \subseteq \text{post}^*(I) \). Let \( R_{\text{sim}} \in \text{QO}(\Sigma) \) and \( P_{\text{sim}} \in \text{Part}(\Sigma) \) be, resp., the simulation qo and partition w.r.t. \( R_i \). Let \( P \triangleq \{ y \in \Sigma \mid R(y) = R(x) \}_{x \in \Sigma} \in \text{Part}(\Sigma) \). Then:

\[
\begin{align*}
P_{\text{sim}}^{\text{post}}(I) & = R^*, \quad (1.a) \\
P_{\text{sim}}^{\text{post}}(I) & \subseteq \{ B \in P \mid R(B) \cap \sigma \neq \emptyset \}. \quad (1.b)
\end{align*}
\]

The following example shows that the containment of (1.b) may be strict.
Example 4.2. Let us consider the system \((\Sigma = \{1, 2\}, I = \{1\}, L, \rightarrow = \{(1, 1)\})\), where \(L\) is a singleton. Moreover, fix \(\sigma_i = \emptyset\) and \(R_i = \{1, 2\} \times \{1, 2\}\). Here, we have that \(\text{post}^*(I) = \{1\}\), \(R_{\text{sim}}(1) = \{1\}\), \(R_{\text{sim}}(2) = \{1, 2\}\), so that \(P_{\text{sim}} = \{(1, 1), (2, 2)\}\). Algorithm 1 on input \(R_i\), \(\sigma_i\) returns \(R = R_{\text{sim}}, \sigma = \{1\}\). Thus, the containment (1.b) turns out to be strict since \([1, 1] \not\subset [1, 1], [2, 2]\). ♦

Next we show that Algorithm 1 always terminates on finite state systems:

**Theorem 4.3 (Termination of Algorithm 1).** Let \(G = (\Sigma, I, L, \rightarrow)\) with \(|\Sigma| \in \mathbb{N}\), \(R_i \in \text{QO}(\Sigma)\) and \(\sigma_i \subseteq \text{post}^*(I)\). Then, Algorithm 1 terminates on input \(G, R_i, \) and \(\sigma_i\).

While termination holds for finite state systems, Algorithm 1 may terminate on infinite state systems as well as shown in the next example.

Example 4.4. Consider the following transition system with infinitely many states where \(I = \{0\}\) is the initial state and \(L\) is a singleton. The transitions are depicted below where the boxes gather states sharing the same principal in \(R_i\). We fix the inputs of Algorithm 1 to be \(\sigma_i = \emptyset\) and \(R_i\), the initial \(\text{qo}\), to be given by \(R_i(0) = \{0\}, R_i(1) = \{0, 1\},\) and \(R_i(n) = [-\infty, -1]\) for \(n < 0\). The simulation quasiorder is: \(R_{\text{sim}}(0) = \{0\}, R_{\text{sim}}(1) = \{0, 1\},\) and \(R_{\text{sim}}(n) = [-\infty, n]\) for each \(n > 0\). Notice that \(R_{\text{sim}}\) has infinitely many principals.

![Diagram of the transition system](image)

Executing Algorithm 1 we get \(\sigma = \{0, 1\}\) after two \(\text{Search}\) iterations. At this point \(V = \emptyset\) and \(U = \emptyset\), and the algorithm returns with the correct result.

Now consider instead the process of refining each principal \(R_i(x)\) such that \(R_i(x) \neq R_{\text{sim}}(x)\). This process, which converges to \(R_{\text{sim}}\), does not terminate after finitely many steps since infinitely many principals need to be refined. ♦

Let us observe that if Algorithm 1 is called with \(\sigma_i = \text{post}^*(I)\), then the set \(U\) will be empty at each iteration, so that \(\text{Search}\) never executes. Based on this observation, we define Algorithm 2 as the partial evaluation of Algorithm 1 under such input. Moreover, Algorithm 2 differs from Algorithm 1 in that we require the input relation \(R_i\) to be reflexive but not necessarily transitive. The rationale is that, later, we use Algorithm 2 as a subroutine with, as input, a relation \(R_i\) that sits in between a quasiorder \(R_{\text{qo}}\) and the simulation quasiorder \(R_{\text{sim}}\) induced by \(R_{\text{qo}}\). Formally we have that the inclusions \(R_{\text{sim}} \subseteq R_i \subseteq R_{\text{qo}}\) hold. Note that \(R_i\) is reflexive since \(R_{\text{sim}}\) is but \(R_i\) need not be transitive. Under
these assumptions on the inputs, it is easy to see that Algorithm 2 inherits correctness and termination results of Theorems 4.1 and 4.3.

Algorithm 2: Sound Refinement Algorithm (SymRef)

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

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

5 A Complete Online Reduction Algorithm

Algorithm 1 is sound but, as shown in Example 4.2, in general not complete. In order to achieve completeness, we perform several key changes to Algorithm 1 resulting in Algorithm 3. First we modify the definitions of \( U \) and \( V \) at lines 5 and 6. Informally speaking, the changes made to the definitions of \( U \) and \( V \) reflects the difference between (2) and (3). Moreover, the algorithm keeps track of a subset of \( U \), denoted by \( U_{\text{bad}} \), that contains states on which executing a Search iteration can never expand \( \sigma \). Furthermore, we added an Expand guarded command in the nondeterministic choice whose role is to guarantee progress by adding new elements to \( \sigma \). The Expand guarded command also contains a return statement invoking Algorithm 2 as a subroutine whenever \( \sigma \) coincides with the set \( \text{post}^*(I) \) of reachable states. It is worth recalling that results in Section 3.2 suggest that computing \( \text{post}^*(I) \) is, in general, unavoidable.

The main feature of Algorithm 3 is that if it terminates with output \( \langle R, \sigma \rangle \) then it induces a partition such that the blocks having a nonempty intersection with \( \sigma \) are in a 1-to-1 correspondence with the reachable blocks of \( P_{\text{sim}} \), and they split every reachable state consistently with \( P_{\text{sim}} \) (3.1 below). Note that one block \( B \) in \( P^* \) might be strictly contained in the corresponding block \( P_{\text{sim}}(B) \) of \( P_{\text{sim}} \), but the algorithm ensures that \( P_{\text{sim}}(B) \setminus B \cap \text{post}^*(I) = \emptyset \) (3.3 below).

On the other hand, Algorithm 3 is also complete for the reachable principals of the simulation qo \( R_{\text{sim}} \) where reachability is defined according to 6 (6.1 below).
Algorithm 3: Complete Algorithm

**Input:** A transition system $G = (\Sigma, I, L, \rightarrow)$, an initial $q_o \in QO(\Sigma)$ and an initial set $I \subseteq \sigma \subseteq post^*(I)$.

1. **forall** $x \in \Sigma$ do
   \[ \wp(\Sigma) \ni R(x) := R_i(x); \]
2. $\wp(\Sigma) \ni \sigma := \sigma_i$;
3. $\wp(\Sigma) \ni U_{bad} := \emptyset$;
4. while true do
   // Inv$_1$: $(\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x))$
   // Inv$_2$: $(\sigma_i \subseteq \sigma \subseteq \text{post}^*(I))$
   // Inv$_3$: $(\forall x \in \Sigma. x \in R(x))$
   // Inv$_4$: $(U_{bad} \subseteq U)$
   5. $U := \{ x \in \Sigma \mid \exists s \in \sigma. R(x) = R(s), R(x) \cap \text{post}(\sigma) \neq \emptyset \}$;
   6. $V := \{ (a, x, x') \in L \times \Sigma^2 \mid \exists s \in \sigma. R(x) = R(s), x \xrightarrow{a} x', R(x') \not\in \text{pre}_a(R(x')) \}$;
7. **nif** $(U \setminus U_{bad} \neq \emptyset)$ **→ Search** :
   8. choose $x \in U \setminus U_{bad}$;
   9. $S := (R(x) \cap \text{post}(\sigma)) \setminus \sigma$;
   10. **if** $S \neq \emptyset$ **then**
    11. choose $s \in S$;
    12. $\sigma := \sigma \cup \{ s \}$;
    13. $U_{bad} := \emptyset$;
   14. **else**
7. $U_{bad} := U_{bad} \cup \{ x \}$;
15. **end**
16. $(V \neq \emptyset)$ **→ Refine** :
17. choose $(a, x, x') \in V$;
18. $S := \text{pre}_a(R(x'))$;
19. $R(x) := R(x) \cap S$;
20. $U_{bad} := \emptyset$;
21. $(U = U_{bad} \neq \emptyset \land V = \emptyset)$ **→ Expand** :
22. **if** $\text{post}(\sigma) \subseteq \sigma$ **then**
    // Run Algorithm 2 starting from the current $R$ and $\sigma$
    return $\text{SymRef}(R, \sigma)$;
23. **else**
24. $\sigma := \sigma \cup \text{post}(\sigma)$;
25. $U_{bad} := \emptyset$;
26. **end**
27. $(U = \emptyset \land V = \emptyset)$ **→ return** $(R, \sigma)$;
28. **end**
29. **end**
30. **end**
31. **fin**
Theorem 5.1 (Correctness of Algorithm 3). Let \( \langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \phi(\Sigma) \) be the output of Algorithm 3 on input \( G \) with \(|\Sigma| \in \mathbb{N}, R_i \in \text{QO}(\Sigma), I \subseteq \sigma_i \subseteq \text{post}^*(I) \). Let \( R_{\text{sim}} \in \text{QO}(\Sigma) \) and \( P_{\text{sim}} \in \text{Part}(\Sigma) \) be, resp., the simulation \( qo \) and partition w.r.t. \( R_i \). Let \( P \equiv \{ y \in \Sigma \mid R(y) = R(x) \}_{x \in \Sigma} \in \text{Part}(\Sigma) \). Then:

\[
\begin{align*}
\{ R_{\text{sim}}(x) \mid x \in \text{post}^*(I) \} &= \{ R(x) \mid x \in \sigma \}, \\
\{ B \cap \text{post}^*(I) \mid B \in P_{\text{sim}}^\text{post}^*(I) \} &= \{ B \cap \text{post}^*(I) \mid B \in P^\sigma \}, \\
P_{\text{sim}}^\text{post}^*(I) &= \{ P_{\text{sim}}(B) \mid B \in P^\sigma \}.
\end{align*}
\]

The following example exhibits the importance of the \textit{Expand} guarded command and of the refinement subroutine (Algorithm 2) in ensuring termination.

Example 5.2. Consider the following finite transition system, where \( I = \{0\} \), \( L \) is a singleton, \( \sigma_i = I \), and the initial \( qo \) \( R_i \) is given by \( R_i(0) = R_i(1) = \{0, 1\}, R_i(2) = \{2\} \) and \( R_i(3) = \{2, 3\} \). The transitions are depicted below where the boxes gather states sharing the same principal in \( R_i \). The simulation quasiorder \( R_{\text{sim}} \) is such that \( R_{\text{sim}}(0) = \{0, 1\}, R_{\text{sim}}(1) = \{1\}, R_{\text{sim}}(2) = \{2\} \) and \( R_{\text{sim}}(3) = \{3\} \), meaning that states 0 and 1 are split in \( P_{\text{sim}} \).

![Transition System Diagram]

Executing Algorithm 3 we find that \( \sigma = \{0, 2\} \) after the first \textit{Search} iteration. At this point \( V = \emptyset \) and \( U = \{3\} \), and 3 will be inserted into \( U_{\text{bad}} \) after a further \textit{Search} iteration. Executing an \textit{Expand} step is then unable to enlarge \( \sigma \) since \( \text{post}(\sigma) \subseteq \sigma \), so that the algorithm will execute the refinement subroutine at line 22 which results in \( R(1) \) to be \( \{1\} \), hence splitting 0 and 1 in \( P \).

As aforementioned, Algorithm 3 terminates on finite state systems.

Theorem 5.3 (Termination of Algorithm 3). Let \( G = (\Sigma, I, L, \rightarrow) \) with \(|\Sigma| \in \mathbb{N}, I \subseteq \sigma_i \subseteq \text{post}^*(I) \) and \( R_i \in \text{QO}(\Sigma) \). Then, Algorithm 3 terminates on input \( G, R_i \) and \( \sigma_i \).

6 Experimental Results

We developed in Python a prototype implementation of the algorithms presented in this paper and performed experiments to assess the impact of the proposed reduction approach. To the best of our knowledge, no publicly available benchmark for LY minimization algorithm [24] exists (actually, we found no publicly available implementation of LY), thus we had to determine some appropriate
benchmarks for our experimental evaluation. We carried out our experiments over a subset of benchmarks from the BEEM database for explicit model checkers \cite{30}. The selected subset of benchmarks consists of mutual exclusion protocols that we purposely modified to increase the number of unreachable states. Unreachable states were added to the model by performing an “unrolling” of the protocol loop and then by picking an initial state after the unrolled protocols. The rationale behind this modification is that, typically, benchmarks available online as explicit transition systems have no unreachable states and, often, are strongly connected, which means that moving the initial state has no effect on the number of unreachable states. The selected benchmarks were then converted to the input format of our prototype, the BA format \cite{1}, via conversion utilities from the toolkits Spot \cite{13} (from the DVE format of the BEEM benchmarks to the Hanoi Omega-Automata Format) and ROLL \cite{25} (from the Hanoi Omega-Automata Format to BA). We used the well-known mCRL2 toolset \cite{7} to compute the bisimulation quotient $P_{\text{bis}}$.

Our results are summarized in Table 1 where the initial equivalence/quasiorder is set to $R_I = \Sigma \times \Sigma$, and $\sigma_I$ to the empty set. Following their definitions, we infer that the following inequalities must hold for the entries of Table 1:

$$
\Sigma \geq \text{post}^*(I) \geq P_{\text{bis}} \geq P_{\text{sim}} \geq P_{\text{r1}} = P_{\text{post}^*(I)} \geq P_{\text{r1}}.
$$

| Protocol | $\rightarrow$ | $\Sigma$ | $\text{post}^*(I)$ | $P_{\text{bis}}$ | $P_{\text{sim}}$ | $P_{\text{post}^*(I)}$ | $P_{\text{sim}}^*$ | $P_{\text{r1}}$ |
|----------|---------------|-----------|-------------------|----------------|----------------|-------------------|----------------|----------------|
| bakery.1 | 2917          | 1628      | 1443              | 1032           | 487            | 867               | 396            | 396            |
| bakery.2 | 2453          | 1340      | 1083              | 894            | 765            | 690               | 638            | 638            |
| fischer.1| 2748          | 1254      | 634               | 1218           | 452            | 616               | 295            | 295            |
| mcs.1    | 31645         | 11839     | 7597              | 1000           | 524            | 447               | 447            |                |

These results show that the number of reachable blocks can diminish significantly if we consider simulation equivalence w.r.t. bisimulation (the reduction produces less than half as many blocks for bakery.1 and fischer.1 examples). Moreover, observe that for Algorithm 1 the columns $P_{\text{r1}}$ and $P_{\text{post}^*(I)}$ coincide on all examples, meaning that the inclusion of (1.b) is as tight as possible (it holds as equality), and, therefore, no imprecision is introduced through approximation. Of course, it is worth pointing out that the set of benchmarks is fairly small and that every transition system is a mutual exclusion protocol that has been unrolled.
7 Conclusion and Future Work

We introduced and proved the soundness of several algorithms with different trade-offs, assumptions and limitations for the online simulation reduction problem, which discloses some fundamental differences in decidability and complexity w.r.t. to the bisimulation case studied by Lee and Yannakakis [24]. To the best of our knowledge, this is the first investigation of this problem. Algorithm 1 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—transition systems with infinitely many states.

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

– To investigate the reduction problem specifically for transition systems with infinitely many states. Note that our algorithms are already applicable to infinite transition systems, and that Appendix C includes correctness results for the infinite case under some assumption.
– To define symbolic algorithms where principals are represented with a partition-relation pair, as done in prior work on simulation algorithms [21,33].
– 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 [8].
– To study algorithmic improvements of Algorithms 1 and 3, for instance, by leveraging the data structures such as queue and stacks used by Lee and Yannakakis’ online algorithm [24] for bisimulation.
– To establish runtime guarantees of the algorithms as the ones stated by Lee and Yannakakis [24] Section 3.

References

1. The BA format. [http://www.languageinclusion.org/doku.php?id=tools#the_ba_format](http://www.languageinclusion.org/doku.php?id=tools#the_ba_format)
2. Alur, R., Henzinger, T.A.: Computer-Aided Verification (1999), [https://www.cis.upenn.edu/~alur/CAVBook.pdf](https://www.cis.upenn.edu/~alur/CAVBook.pdf) chapter 4: Graph Minimization
3. Bensalem, S., Bouajjani, A., Loiseaux, C., Sifakis, J.: Property preserving simulations. In: Proc. of the Fourth International Workshop on Computer Aided Verification, CAV’92. pp. 260–273. LNCS, Springer (1992). [https://doi.org/10.1007/3-540-56496-9_21](https://doi.org/10.1007/3-540-56496-9_21)
4. Bloom, B., Paige, R.: Transformational design and implementation of a new efficient solution to the ready simulation problem. Sci. Comput. Program. 24(3), 189–220 (1995). [https://doi.org/10.1016/0167-6423(95)00003-B](https://doi.org/10.1016/0167-6423(95)00003-B)
5. Bouajjani, A., Fernandez, J.C., Halbwachs, N.: Minimal model generation. In: Proc. of the International Workshop on Computer-Aided Verification, CAV’91. pp. 197–203. LNCS, Springer (1991). [https://doi.org/10.1007/BFb0023733](https://doi.org/10.1007/BFb0023733)
6. Bouajjani, A., Fernandez, J.C., Halbwachs, N., Raymond, P., Ratel, C.: Minimal state graph generation. Science of Computer Programming 18(3), 247–269 (1992). [https://doi.org/10.1016/0167-6423(92)90018-7](https://doi.org/10.1016/0167-6423(92)90018-7)
7. Bunte, O., Groote, J.F., Keiren, J.J.A., Laveaux, M., Neele, T., de Vink, E.P., Wesseling, W., Wijs, A., Willemsse, T.A.C.: The mCRL2 toolset for analysing concurrent systems. In: Proc. of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2019, pp. 21–39. Springer (2019). https://doi.org/10.1007/978-3-030-17465-1_2
8. Bustan, D., Grumberg, O.: Simulation-based minimization. ACM Trans. Comput. Log. 4(2), 181–206 (2003). https://doi.org/10.1145/635499.635502
9. Cécé, G.: Foundation for a series of efficient simulation algorithms. In: Proc. of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005069
10. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Progress on the state explosion problem in model checking. In: Informatics - 10 Years Back. 10 Years Ahead. pp. 176–194. LNCS, Springer (2001). https://doi.org/10.1007/3-540-44577-3_12
11. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R.: Handbook of Model Checking. Springer, 1st edn. (2018)
12. Crafa, S., Ranzato, F., Tapparo, F.: Saving space in a time efficient simulation algorithm. Fundam. Informaticae 108(1-2), 23–42 (2011). https://doi.org/10.3233/FI-2011-412
13. Duret-Lutz, A., Renault, E., Colange, M., Renkin, F., Aisse, A.G., Schellhuber-Caissier, P., Medioni, T., Martin, A., Dubois, J., Gillard, C., Lauko, H.: From Spot 2.0 to Spot 2.10: What’s new? In: Proc. of the 34th International Conference on Computer Aided Verification, CA V’22. pp. 174–187. LNCS, Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_9
14. Fisler, K., Vardi, M.Y.: Bisimulation minimization and symbolic model checking. Formal Methods Syst. Des. 21(1), 39–78 (2002). https://doi.org/10.1023/A:1016091902809
15. Gentilini, R., Piazza, C., Policriti, A.: From bisimulation to simulation: Coarsest partition problems. Journal of Automated Reasoning 31(1), 73–103 (2003). https://doi.org/10.1023/A:1027328830731
16. Glabbeek, R.v., Ploeger, B.: Correcting a space-efficient simulation algorithm. In: Proc. of the International Conference on Computer Aided Verification, CA V’08. pp. 517–529. Springer (2008). https://doi.org/10.1007/978-3-540-70545-1_49
17. Grumberg, O., Long, D.E.: Model checking and modular verification. In: Proc. of the 2nd International Conference on Concurrency Theory, CONCUR’91. pp. 250–265. LNCS, Springer (1991). https://doi.org/10.1007/3-540-54430-5_93
18. Grumberg, O., Long, D.E.: Model checking and modular verification. ACM Transactions on Programming Languages and Systems (TOPLAS) 16(3), 843–871 (1994). https://doi.org/10.1145/177492.177725
19. Gulavani, B.S., Henzinger, T.A., Kannan, Y., Nori, A.V., Rajamani, S.K.: SYNERGY: a new algorithm for property checking. In: Proc. of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006. pp. 117–127. ACM (2006). https://doi.org/10.1145/1181775.1181790
20. Habib, M., Paul, C., Viennot, L.: Partition refinement techniques: An interesting algorithmic tool kit. Int. J. Found. Comput. Sci. 10(2), 147–170 (1999). https://doi.org/10.1142/S0129054199000125
21. Henzinger, M., Henzinger, T., Kopke, P.: Computing simulations on finite and infinite graphs. In: Proc. of IEEE 36th Annual Foundations of Computer Science, FOCS’95. pp. 453–462 (1995). https://doi.org/10.1109/SFCS.1995.492576
22. Kucera, A., Mayr, R.: Simulation preorder over simple process algebras. Inf. Comput. 173(2), 184–198 (2002). https://doi.org/10.1006/inco.2001.3122
23. Kucera, A., Mayr, R.: Why is simulation harder than bisimulation? In: Proc. of the 13th International Conference on Concurrency Theory, CONCUR 2002. pp. 594–610. LNCS, Springer (2002). https://doi.org/10.1007/3-540-45694-5_39

24. Lee, D., Yannakakis, M.: Online Minimization of Transition Systems. In: Proc. of the 24th Annual ACM Symposium on Theory of Computing, STOC ’92. pp. 264–274. ACM (1992). https://doi.org/10.1145/129712.129738

25. Li, Y., Sun, X., Turrini, A., Chen, Y., Xu, J.: ROLL 1.0: ω-regular language learning library. In: Proc. of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2019. pp. 365–371. LNCS, Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_23

26. Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., Bensalem, S.: Property preserving abstractions for the verification of concurrent systems. Formal Methods Syst. Des. 6(1), 11–44 (1995). https://doi.org/10.1007/BF01384313

27. Majumdar, R., Ozay, N., Schmuck, A.K.: On abstraction-based controller design with output feedback. In: Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control, HSCC 2020. pp. 1–11. ACM (2020). https://doi.org/10.1145/3365365.3382219

28. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall Series in Automatic Computation. Prentice-Hall (1967)

29. Pasareanu, C.S., Pelánek, R., Visser, W.: Concrete model checking with abstract matching and refinement. In: Proc. 17th International Conference on Computer Aided Verification, CAV 2005. pp. 52–66. LNCS, Springer (2005). https://doi.org/10.1007/11513988_7

30. Pelánek, R.: BEEM: Benchmarks for explicit model checkers. In: Model Checking Software, pp. 263–267. Springer (2007). https://doi.org/10.1007/978-3-540-73370-6_17

31. Ranzato, F.: A more efficient simulation algorithm on Kripke structures. In: Proc. of the 38th International Symposium on Mathematical Foundations of Computer Science 2013, MFCS 2013. pp. 753–764. LNCS, Springer (2013). https://doi.org/10.1007/978-3-642-40313-2_66

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

33. Ranzato, F., Tapparo, F.: A new efficient simulation equivalence algorithm. In: Proc. of the 22nd IEEE Symposium on Logic in Computer Science, LICS 2007. pp. 171–180. IEEE Computer Society (2007). https://doi.org/10.1109/LICS.2007.8

34. Ranzato, F., Tapparo, F.: An efficient simulation algorithm based on abstract interpretation. Information and Computation 208(1), 1–22 (2010). https://doi.org/10.1016/j.ic.2009.06.002

35. Sipser, M.: Introduction to the Theory of Computation. Thomson Course Technology, 2nd ed edn. (2006)

36. Tan, L., Cleaveland, R.: Simulation revisited. In: Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001. pp. 480–495. LNCS, Springer (2001). https://doi.org/10.1007/3-540-45319-9_33
A Auxiliary Algorithms

Algorithm 4: Sim

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

1 forall $x \in \Sigma$ do $\varphi(\Sigma) \ni R(x) := R_i(x);
2$ while $\exists x, x' \in \Sigma, a \in L, x \rightarrow x'$ and $R(x) \not\subseteq \text{pre}_a(R(x'))$ do
   // Inv1: $(\forall x \in \Sigma. R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x))$
3 $R(x) := R(x) \cap \text{pre}_a(R(x'));
4$ end
5 return $R$;

Assumption 1. Let $y \in R_i(x) \setminus R_{\text{sim}}(x)$ for some $x \in \Sigma$, then there exists an execution (not necessarily terminating) of Sim over input $R_i$ such that, after a finite number of iterations it holds $y \not\in R(x)$.

B Proofs

Theorem 4.1 (Correctness of Algorithm 1). Let $(R, \sigma) \in \text{Rel}(\Sigma) \times \varphi(\Sigma)$ be the output of Algorithm 7 on input $G$ with $|\Sigma| \in \mathbb{N}, R_i \in \text{QO}(\Sigma)$ and $\sigma_i \subseteq \text{post}^*(I)$. Let $R_{\text{sim}} \in \text{QO}(\Sigma)$ and $P_{\text{sim}} \in \text{Part}(\Sigma)$ be, resp., the simulation $\text{qo}$ and partition w.r.t. $R_i$. Let $P \triangleq \{ y \in \Sigma \mid R(y) = R(x) \}_{x \in \Sigma} \in \text{Part}(\Sigma)$. Then:

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

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

Proof. The following facts on the output tuple $(R, \sigma)$ hold:

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

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

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

Let Sim be the basic simulation algorithm as recalled in Algorithm 1. By (iii), $R_{\sim} \subseteq R \subseteq R_i$. Thus, $\text{Sim}(R) = R_{\sim}$ because $R_{\sim} = \text{Sim}(R_{\sim}) \subseteq \text{Sim}(R) \subseteq \text{Sim}(R_i) = R_{\sim}$. Assume, by contradiction, that $S \neq \{R(x) \in \phi(\Sigma) \mid R(x) \cap \sigma \neq \emptyset, R(x) \neq R_{\sim}(x)\} \neq \emptyset$, so that each $R(x) \in S$, will be refined at some iteration of $\text{Sim}(R)$. Then, let $R(x) \in S$ be the first principal in $S$ such that $R(x)$ is refined by Sim at some iteration it whose current relation is $R'$. Thus, each principal $R(z) \in S$ is such that $R(z) = R'(z)$ because no principal in $S$ was refined by Sim before this iteration it. Let $R'(y) \subseteq R(y)$ be the principal of $R'$ used by Sim in this iteration it to refine $R(x)$, so that $x \rightarrow y$, $R'(x) = R(x) \not\subseteq \text{pre}_a(R'(y))$. We have $R(x) \cap \sigma \neq \emptyset$, $x \rightarrow y$ and $V = \emptyset$ entail $R(x) \subseteq \text{pre}_a(R'(y))$. Hence, $R(x) \cap \sigma \neq \emptyset$ implies $R(y) \cap \text{post}_a(\sigma) \neq \emptyset$, so that $U = \emptyset$ entails $R(y) \cap \sigma \neq \emptyset$. If $R(y) \not\subseteq S$ then $R(y) \cap \sigma \neq \emptyset$ implies $R(y) = R_{\sim}(y) \subseteq R'(y) \subseteq R(y)$, so that $R'(y) = R(y)$ holds. If $R(y) \subseteq S$ then, since $R(x)$ is the first principal in $S$ to be refined by Sim, we have that $R'(y) = R(y)$ holds. Thus, in both cases, $R'(y) = R(y)$ must hold, so that $R(x) \not\subseteq \text{pre}_a(R'(y))$. Moreover, $R(x) \cap \sigma \neq \emptyset$, $x \rightarrow y$ and $V = \emptyset$ imply $R(x) \subseteq \text{pre}_a(R'(y))$, which is therefore a contradiction. Thus, $S = \emptyset$, and, in turn, $R(x) = R_{\sim}(x)$.

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

Let us now show the equality (1a).

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

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

Let us now prove (1b). Let $x \in \Sigma$ such that $P_{\sim}(x) \cap \text{post}^*(I) \neq \emptyset$. By definition of $P$, we have that for all $x \in \Sigma$, $P(x) = \{y \in \Sigma \mid R(x) = R(y)\}$. Since $P_{\sim}(x) \subseteq R_{\sim}(x)$, we have that $P_{\sim}(x) \cap \text{post}^*(I) \neq \emptyset$. Let $y \in P_{\sim}(x)$.

Observe that $P_{\sim}(x) = \{z \in \Sigma \mid R_{\sim}(x) = R_{\sim}(z)\}$. Thus, $R_{\sim}(x) = R_{\sim}(y)$, so that $R_{\sim}(y) \cap \text{post}^*(I) \neq \emptyset$. Thus, by (1a), $R_{\sim}(y) = R(y)$ and $R(y) \cap \sigma \neq \emptyset$.

In particular, $R(x) = R_{\sim}(x) = R_{\sim}(y) = R(y)$. Therefore, $P_{\sim}(x) \subseteq \{y \in \Sigma \mid R(x) = R(y)\}$. 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 (1a), $R_{\sim}(z) = R(z)$. Hence, $R_{\sim}(z) = R(z) = R_{\sim}(x)$, thus proving that $P(x) \subseteq P_{\sim}(x)$, and therefore $P(x) = P_{\sim}(x)$ holds. Moreover, $R(x) \cap \sigma \neq \emptyset$, thus proving (1b).

Theorem 3.3 (Termination of Algorithm 1). Let $G = (\Sigma, I, L, \rightarrow)$ with $|\Sigma| \in \mathbb{N}$, $R_i \in \mathcal{Q}(\Sigma)$ and $\sigma_i \subseteq \text{post}^*(I)$. Then, Algorithm 1 terminates on input $G$, $R_i$, and $\sigma_i$. \hfill $\square$
Proof. We first observe that each Search iteration adds some new state to \( \sigma \) through the update at line 10. Thus, since \( \Sigma \) has finitely many elements, Algorithm 11 will always execute a finite number of Search iterations. Similarly, executing the Refine block is guaranteed to remove some state from at least one principal of the current relation, and since each principal has an initial finite number of elements, the overall number of Refine iterations is also finite. Therefore, since every iteration of the while-loop either executes a Search or a Refine, the algorithm terminates after a finite number of iterations. Finally, we observe that each iteration computes a finite number of operations, so that this completes the proof.

\[ \top \]

Theorem 5.1 (Correctness of Algorithm 9). Let \( \langle R, \sigma \rangle \in \text{Rel}(\Sigma) \times \varphi(\Sigma) \) be the output of Algorithm 9 on input \( G \) with \( |\Sigma| \in \mathbb{N}, R_i \in \text{QO}(\Sigma), I \subseteq \sigma_i \subseteq \text{post}^*(I) \). Let \( R_{\text{sim}} \in \text{QO}(\Sigma) \) and \( P_{\text{sim}} \in \text{Part}(\Sigma) \) be, resp., the simulation \( qo \) and partition w.r.t. \( R_i \). Let \( P \triangleq \{y \in \Sigma \mid R(y) = R(x)\}_{x \in \Sigma} \in \text{Part}(\Sigma) \). Then:

\[
\begin{align*}
\{ R_{\text{sim}}(x) \mid x \in \text{post}^*(I) \} &= \{ R(x) \mid x \in \sigma \}, \tag{3.a} \\
\{ B \cap \text{post}^*(I) \mid B \in \text{P}^{\text{post}^*(I)} \} &= \{ B \cap \text{post}^*(I) \mid B \in \text{P}^* \}, \tag{3.b} \\
\text{P}^{\text{post}^*(I)} &= \{ P_{\text{sim}}(B) \mid B \in \text{P}^* \} . \tag{3.c}
\end{align*}
\]

Proof. We first observe that at termination \( V = \emptyset \) and one of the two following holds:

\[
\begin{align*}
U &= \emptyset \tag{4} \\
U &\neq \emptyset \land (\text{post}(\sigma) \subseteq \sigma) \tag{5}
\end{align*}
\]

Corresponding to, respectively, the case where the algorithm executes line 25 or 30. Moreover, in the case \( U \neq \emptyset \), we observe that \( \text{post}(\sigma) \subseteq \sigma \), together with \( I \subseteq \sigma \) allows us to conclude that \( \text{post}^*(I) \subseteq \sigma \) by fixpoint definition of \( \text{post}^*(I) \), and Inv2 implies \( \sigma = \text{post}^*(I) \). Moreover, by correctness of Algorithm 2 we get that, at line 25 \( \sigma \subseteq \sigma' \subseteq \text{post}^*(I) \) and thus \( \sigma = \sigma' = \text{post}^*(I) \).

We now show that the following facts on the output pair \( \langle R, \sigma \rangle \) of Algorithm 9 hold:

\[
\begin{enumerate}
\item [(i)] For all \( x \in \Sigma \), \( R_{\text{sim}}(x) \subseteq R(x) \).
\end{enumerate}
\]

This holds because the Refine block of Algorithm 9 is always correct, so that, for all \( x \in \Sigma \), \( R_{\text{sim}}(x) \subseteq R(x) \subseteq R_i(x) \) holds for every iteration of Algorithm 9. Moreover, correctness of Algorithm 2 ensures that line 25 preserves this property.

\[
\begin{enumerate}
\item [(ii)] \( x \in \text{post}^*(I) \Rightarrow \exists s \in \sigma. R(x) = R(s) \).
\end{enumerate}
\]

We distinguish the two possible cases at termination:

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

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

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

We distinguish the two possible cases at termination:

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

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

By (i), \( R_{\text{sim}}(z) \subseteq R(z) \subseteq R_i(z) \), for all \( z \in \Sigma \). Moreover, \( \text{Sim}(R) = R_{\text{sim}} \) because \( R_{\text{sim}} = \text{Sim}(R_{\text{sim}}) \subseteq \text{Sim}(R) \subseteq \text{Sim}(R_i) = R_{\text{sim}} \). Let us assume, by contradiction, that \( S = \{ R(x) | x \in \Sigma, R(x) \neq R_{\text{sim}}(x), \exists s \in \sigma. R(x) = R(s) \} \neq \emptyset \), so that every principal in \( S \) will be refined at some iteration of Sim on input \( R \). Let \( R(x) \in S \) be the first principal in \( S \) to be refined by Sim at some iteration \( i \) whose current relation is \( R' \) with \( R' \subseteq R \), so that each principal in \( S \) is a principal for \( R' \) since no principal in \( S \) was refined before this iteration \( i \). Let \( R'(y) \subseteq R(y) \) be the principal of \( R' \) used in this iteration \( i \) to refine \( R(x) \), so that \( x \rightarrow y, R'(x) = R(x) \not\subseteq \text{pre}_a(R'(y)) \).

We have that \( x \rightarrow y \), along with \( \exists s \in \sigma. R(s) = R(x) \) and \( V = \emptyset \) entails \( R(x) \subseteq \text{pre}_a(R(y)) \). Hence, by reflexivity \( s \in R(s) = R(x) \subseteq \text{pre}_a(R(y)) \) implies \( R(y) \cap \text{post}(\sigma) \neq \emptyset \), so that \( U = \emptyset \) entails \( \exists s' \in \sigma. R(y) = R(s') \).

Now, either \( R(y) \in S \), which implies \( R(y) = R_{\text{sim}}(y) \) and therefore \( R(y) = R'(y) \), or \( R(y) \in S \) and since no principal in \( S \) was refined before, \( R(y) = R'(y) \).

This lets us conclude that \( R(x) \not\subseteq \text{pre}_a(R'(y)) = \text{pre}_a(R(y)) \), which is a contradiction to \( R(x) \subseteq \text{pre}_a(R(y)) \). Thus \( S = \emptyset \) and, in turn \( R(x) = R_{\text{sim}}(x) \).

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

We first show that

\[
(\exists s \in \sigma. R(s) = R(x)) \Rightarrow P(x) \subseteq P_{\text{sim}}(x). \tag{6}
\]

Consider an element \( y \in P(x) \), then by definition of \( P \) it holds \( R(y) = R(x) \) and thus \( R(y) = R(s) \), moreover, by (iii) we get \( R_{\text{sim}}(y) = R(y) = R(x) = R_{\text{sim}}(x) \) and thus by definition of \( P_{\text{sim}} \) it follows that \( y \in P_{\text{sim}}(x) \), meaning \( P(x) \subseteq P_{\text{sim}}(x) \), which proves \( 6 \). The inclusion \( P(x)^{\cap} \text{post}^*(I) \subseteq P_{\text{sim}}(x)^{\cap} \text{post}^*(I) \) follows by definition from \( 6 \). On the other hand, pick some \( y \in P_{\text{sim}}(x)^{\cap} \text{post}^*(I) \), then by definition of \( P_{\text{sim}} \) we get \( R_{\text{sim}}(y) = R_{\text{sim}}(x) \), by (i), we get that \( y \in \text{post}^*(I) \) entails \( R(y) = R(s') \) for some \( s' \in \sigma \) and (iii) implies both \( R(y) = R_{\text{sim}}(y) = R_{\text{sim}}(x) = R(x) \). Thus by definition of \( P \), \( y \in P(x) \) and therefore \( P_{\text{sim}}(x)^{\cap} \text{post}^*(I) \subseteq P(x)^{\cap} \text{post}^*(I) \).

\((v)\) \( s \in \sigma \Rightarrow P_{\text{sim}}(s) = P_{\text{sim}}(P(s)) \): Take a state \( s \in \sigma \), then it holds that,

by \( 6 \), \( s \in P(s) \subseteq P_{\text{sim}}(s) \), and by definition of additive lifting it holds \( \bigcup_{x \in \{s\}} P_{\text{sim}}(x) \subseteq \bigcup_{x \in P(s)} P_{\text{sim}}(x) \subseteq \bigcup_{x \in P_{\text{sim}}(s)} P_{\text{sim}}(x) \) that, in turn, en-
tails $P_{\text{sim}}(s) \subseteq P_{\text{sim}}(P(s)) \subseteq P_{\text{sim}}(P_{\text{sim}}(s)) = P_{\text{sim}}(s)$, and therefore $P_{\text{sim}}(s) = P_{\text{sim}}(P(s))$.

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

This holds since \( \sigma_i \subseteq \text{post}^*(I) \) and updates at lines 23, 27 preserve this property.

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

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

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

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

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

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

**Theorem 5.3** (Termination of Algorithm 3). Let \( G = (\Sigma, I, L, \rightarrow) \) with \( |\Sigma| \in \mathbb{N} \), \( I \subseteq \sigma_1 \subseteq \text{post}^*(I) \) and \( R_i \in \text{QO}(\Sigma) \). Then, Algorithm 3 terminates on input \( G, R_i \) and \( \sigma_i \).

**Proof.** We first observe that executing the Refine block is guaranteed to remove some state from at least one principal of the current relation, and since each principal has an initial finite number of elements, the overall number of Refine iterations is finite. Then, every Expand iteration either adds some new state to \( \sigma \) through the update at line 27 or it reaches the return statement at line 25 and thus the number of Expand iterations is also finite. Moreover, every Search iteration will either add some new state to \( \sigma \) through the update at line 13 or some new state to \( U_{\text{bad}} \) through the update at line 16. Thus, since \( \Sigma \) has finitely many elements, the algorithm will always execute a finite number of Search iterations which expand \( \sigma \). Finally, the number of Search iterations expanding \( U_{\text{bad}} \) occurring in between two iterations resetting \( U_{\text{bad}} \) to \( \emptyset \) is also finite, since
$U_{bad} \subseteq \Sigma$ and $\Sigma$ is finite, and observing that the iterations executing $U_{bad} := \emptyset$ are, in turn, either Refine iterations, non terminating Expand iterations, or Search iterations expanding $\sigma$, which we have shown to be finitely many allows us to conclude that the total number of Search iterations is also finite.

Therefore, since every iteration of the while-loop executes either a Search, Refine or Expand iteration, the algorithm terminates after a finite number of iterations. Observing that each iteration computes a finite number of operations, and in particular the execution of Algorithm 2 at line 25 is guaranteed to terminate by Theorem 4.3 completes the proof.

\[\square\]

C Extension to Infinite Transition Systems

We now briefly discuss how to extend the proofs for Theorems 4.1 and 5.1 to infinite state systems. We first note that the only points in the two proofs where finiteness of the state space is relied upon are, respectively, point (iv) of Theorem 4.1 and point (iii) of Theorem 5.1; therefore we proceed to show how their common pattern can generalized to transition systems with infinitely many states (in the following we refer to the notation used in both proofs). Observe that, if the input transition system is finite, then the execution of $\text{Sim}$ over the output relation $R$ is guaranteed to terminate, and correctness of $\text{Sim}$ ensures that an iteration $it$ where some element of $\mathcal{S}$ is refined will be eventually reached in finite time, for every execution (regardless of how nondeterminism is resolved).

On the other hand, for transition systems with infinitely many states, an iteration such as $it$ might not exist since $\text{Sim}$ might not terminate on infinite state systems and, moreover, nondeterminism might choose to refine principals only if they are not marked as reachable (that is, $R(x) \cap \sigma = \emptyset$ for Theorem 4.1 and $\exists s \in \sigma. R(x) = R(s)$ for Theorem 5.1) and which are, therefore, not included in $\mathcal{S}$. However, we can use Assumption 1 to avoid this situation altogether since, assuming $\mathcal{S} \neq \emptyset$ (as we do, by contradiction, in the proofs), we can pick a principal $R(z) \in \mathcal{S}$ and some $z' \in R(z) \setminus R_{\text{sim}}(z) \neq \emptyset$ by definition of $\mathcal{S}$. Assumption 1 therefore ensures that there exists an execution where $\text{Sim}$ reaches an iteration $it_1$ where $z'$ is removed from the principal of $z$. Then, let $R_1$ be the current relation at iteration $it_1 + 1$ and define $\mathcal{S}' = \{ R(x) \mid x \in \Sigma, R(x) \in \mathcal{S}, R_1(x) \neq R(x) \}$ to be the set of principals which have been refined by $\text{Sim}$ up to iteration $it_1 + 1$. We find that $\mathcal{S}'$ is nonempty by construction since $R(z) \in \mathcal{S}'$, and that $R_{\text{sim}} \subseteq R_1 \subseteq R$ following Inv1. The proof then proceeds as explained in the respective points (iv) and (iii) by considering the first principal in $\mathcal{S}'$ to be refined and the corresponding iteration $it$, allowing us to get a contradiction which proves that $\mathcal{S} = \emptyset$. 

\[\Box\]