Rank-Based Simulation on Acyclic Graphs

Raffaella Gentilini[1], Carla Piazza[2], and Alberto Policriti[2]

[1] Dip. di Matematica e Informatica, Università di Perugia, Via Vanvitelli 1, Perugia (IT)
[2] Dip. di Matematica e Informatica, Università di Udine, Via Le Scienze 206, Udine (IT).
{raffaella.gentilini}@dmi.unipg.it, {piazza|policriti}@dimi.uniud.it

Abstract. The simulation preorder is widely used both as a behavioral relation in concurrent systems, and as an abstraction tool to reduce the state space in model checking, were memory requirement is clearly a critical issue. Therefore, in this context a simulation algorithm should address both time and space efficiency. In this paper, we rely on the notion of rank to design an efficient simulation algorithm. It turns out that such algorithm outperforms—both in terms of time and in terms of space—the best simulation algorithms in the literature, on the class of acyclic graphs.

1 Introduction

The simulation preorder [12] is a behavioral refinement relation on labeled graphs, widely used as a formal tool supporting the design and the automated reasoning on complex systems. In particular, simulations plays a role in two tasks that are often crucial to guarantee the success of a formal method for system design or computer aided verification: the system *refinement* and the system *abstraction* [10]. In this context, the behavior of a system or a set of programs implementing a collection of cooperating units is naturally modeled as a (labeled) graph, whose nodes describe the possible states and arrows represent actions. Given a specification of a system as a labeled graph G_1, the simulation preorder provides a formal tool for checking wether G_1 is correctly implemented (or refined) by the concrete system G_2. Moreover, the induced equivalence can be used as an abstraction tool to cope with the intricacies buried in the modeling activity and to control the sheer size of the obtained structures. In particular, space requirements underly the notorious state-explosion problem in *model checking* [5], a fully automatic (and quite efficient in time) formal method for verifying finite-state systems[1] with respect to temporal logics specifications. Abstraction methods for model checking are required to be preservative with respect to the logic language used for specifying the properties of the system. An abstraction method is said to be *weakly preservative* for a temporal logic $\mathcal{L}$ if whenever a property p of $\mathcal{L}$ is true in the abstract structure, p holds also in the concrete model. An abstraction method is said to be *strongly preservative* for a temporal logic $\mathcal{L}$ if both true and false $\mathcal{L}$-properties are preserved from the abstract structure to the concrete model. Grumberg et al. [13] proved that the simulation preorder is weakly preservative for ACTL* and ACTL, the universal fragments of

[1] The labeled graphs used to model the system under verification are called *Kripke structures* in the context of Model Checking

the branching temporal logics CTL and CTL* [4], as well as for the universal fragment of the μ-calculus. In [11], it was shown that the simulation equivalence strongly preserves both the universal and the existential fragment of the μ-calculus. As a consequence, it strongly preserves its sublogics ACTL*, ECTL*, ECTL and ACTL, widely used for model checking. The latter preservation results combined with the existence of a number of polynomial algorithms for computing (the maximal) simulation on a labeled graph [2, 3, 6, 10], explains the appealing of simulation-based abstraction methods in model checking, also w.r.t. other popular behavioral refinement relations such as language equivalence and bisimulation [16]. In fact, language equivalence provides strong preservation of linear temporal properties and large reductions, however its complexity is exponential, whereas the complexity of bisimulation and simulation is polynomial [14, 7, 8]. On the other hand, bisimulation has the advantage (w.r.t. simulation and language equivalence) of preserving more expressive logics. However this is also a disadvantage, since the abstract structure is required to be so close to the original model that the reductions allowed are far less powerful.

State of the Art Among the algorithms for computing the simulation preorder, the most well known ones are by Henzinger, Henzinger and Kopke [10], Bloom and Paige [1], Bustan and Grumberg [2, 3], Tan and Cleaveland [6], Gentilini, Piazza, and Policriti [9, 20], and Ranzato and Tapparo [17, 18]. Given a (labelled) graph G with $|V|$ nodes and $|E|$ edges, let $|V_{\equiv_S}|$ be the size of the maximum simulation (equivalence) on G. The algorithm by Ranzato and Tapparo [17] runs in $\mathcal{O}(|E||V_{\equiv_S}|)$ time and $\mathcal{O}(|E||V||V_{\equiv_S}|)$ space. It is the best up-to-date simulation procedure as far as time complexity is concerned. On the other hand, the algorithm in [9] (that originally had a minor flow, subsequently corrected in [20]) has the best up-to-date space complexity—$\mathcal{O}(|V_{\equiv_S}|^2+|V|\log(|V_{\equiv_S}|))$—and runs in $\mathcal{O}(|E||V_{\equiv_S}|^2)$ time. In [18], Ranzato and Tapparo proposed a new simulation algorithm featuring an improvement w.r.t. the space-complexity of their previous procedure, while slightly worsening the time-performance (of a cubic factor w.r.t. $|V_{\equiv_S}|$).

Our Contribution We propose a simulation algorithm that has optimal performances w.r.t. both time and space on acyclic graphs, outperforming [17, 18, 9]. Namely, our algorithm uses $\mathcal{O}(|E||V_{\equiv_S}|)$ time and $O(|V_{\equiv_S}|^2 + |V|\log(|V_{\equiv_S}|))$ bits to compute a simulation preorder on a given acyclic graph. The time/space improvement w.r.t. [17, 18, 9] relies on computing the maximum simulation proceeding *by rank*.

2 Preliminaries

In this section we introduce the basic notations we use in the rest of the paper.

Definition 1. *Let V be a set and $Q \subseteq V \times V$ a binary relation over V:*

- *Q is said to be a* preorder *over V if and only if Q is reflexive and transitive;*
- *Q is said to be a* partial order *over V if and only if Q is reflexive, antisymmetric, and transitive;*
- *Q is said to be* acyclic *if and only if its transitive closure is antisymmetric.*

We will use Q^+ to refer to the transitive closure of Q and Q^* to refer to the reflexive and transitive closure of Q.

Notice that if a relation is acyclic, then it is antisymmetric, while the converse does not hold (unless it is transitive).

Definition 2. *A triple $G = \langle V, E, \Sigma \rangle$ is said to be a* labelled graph *if and only if $G^- = \langle V, E \rangle$ is a finite graph and Σ is a partition over V. We say that two nodes $v_1, v_2 \in V$ have the same* label *if they belong to the same class of Σ.*

An equivalent way to define *labelled graphs* is to use a labelling function $\ell : V \to L$, where L is a finite set of labels (inducing of a partition Σ_L of V). Given a node $v \in V$ we will use $[v]_\Sigma$ (or $[v]$, if Σ is clear from the context) to denote the class of Σ to which v belongs.

Example 1. A *Kripke Structure* is a labelled graph and, vice-versa, each connected labelled graph can be seen as a Kripke Structure in which two worlds satisfy the same set of atomic propositions if and only if their labels are equal.

Definition 3. *Let $G = \langle V, E, \Sigma \rangle$ be a labelled graph. A relation $\leq \subseteq V \times V$ is said to be a* simulation *over G if and only if:*

1. *$v \leq u \to [v]_\Sigma = [u]_\Sigma$;*
2. *$(v \leq u \wedge vEv_1) \to \exists u_1 (uEu_1 \wedge v_1 \leq u_1)$.*

In this case we also say that u simulates *v.*
We say that u and v are sim-equivalent *($u \equiv_s v$) if there exist two simulations $\leq_1$ and $\leq_2$, such that $n \leq_1 m$ and $m \leq_2 n$.*

Notice that a simulation can be neither reflexive nor transitive (e.g. the empty relation is always a simulation), however the reader can easily verify that given an arbitrary simulation its reflexive and transitive closure is always a simulation. A simulation $\leq_s$ over G is said to be *maximal* if for all the simulations $\leq$ over G it holds $\leq \subseteq \leq_s$. Given a labelled graph $G = \langle N, E, \Sigma \rangle$ there always exists a unique maximal simulation $\leq_s$ over G. Moreover $\leq_s$ is a preorder [12].

Example 2. Consider the labelled graph $G = \langle V, E, \Sigma \rangle$ depicted in Figure 1, where $V = \{x, y, z\}$, $E = \{(x, y), (x, z), (y, z)\}$, and $\Sigma = \{\alpha = \{x, y\}, \beta = \{z\}\}$. The maximum simulation preorder on G is given by $I \cup \{(y, x)\}$, where I denotes the identity relation over V.

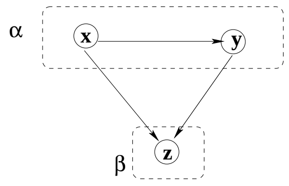

Fig. 1. A labelled graph.

Given a labelled graph G, the *simulation problem* consists in determining the maximum simulation preorder on G, and can be elegantly encoded in terms of a *coarsest partition pair problem* [9]. Such a formulation is the engine of the space efficient procedure in [9, 20] and relies on the fundamental notions of *partition pair* (PP), PP *refinement* and PP *stability*, recalled below.

Definition 4 (Partition Pairs). *Let V be a set. A partition pair on V is a pair $\langle \Sigma, R\rangle$, where Σ is a partition on V and R is a reflexive relation on Σ.*

Given a set V, each preorder relation $\preceq_P$ on V induces a corresponding partition pair $\langle V_{\equiv_P}, P\rangle$, where $\equiv_P$ is the equivalence relation $\equiv_P = \{(u,v) \mid u \prec_P v \wedge v \preceq_P u\}$, and $P = \{(\alpha,\beta) \in V_{\equiv_P} \mid \exists u \in \alpha, \exists v \in \beta.(u \preceq_P v)\}$. In particular, given a labelled graph $G = \langle V, E, \Sigma\rangle$, we denote by $\langle V_{\equiv_S}, S\rangle$ the partition pair on V corresponding to the maximum simulation preorder $\preceq_S$ of G consistent w.r.t. Σ.

Definition 5. *Let $\langle \Sigma, R\rangle, \langle \Pi, P\rangle$ be two partition pairs on V:*

$$\langle \Pi, P\rangle \sqsubseteq \langle \Sigma, R\rangle \Leftrightarrow \Pi \text{ is finer than } \Sigma \text{ and } P \subseteq R(\Pi)$$

where $R(\Pi)$ denotes the relation on Π induced by $R \subseteq \Sigma \times \Sigma$, i.e.:

$$\forall \alpha, \beta \in \Pi((\alpha,\beta) \in R(\Pi) \Leftrightarrow \exists \alpha', \beta'((\alpha',\beta') \in R \wedge \alpha \subseteq \alpha' \wedge \beta \subseteq \beta'))$$

Given two sets of nodes $\alpha, \beta \subseteq V$ we write $\alpha \rightarrow_\exists \beta$ to denote that there exists a node $a \in \alpha$ which reaches a node $b \in \beta$, i.e., $(a,b) \in E$. Similarly, $\alpha \rightarrow_\forall \beta$ denotes that each node in α reaches a node in β.

Definition 6 (Stability). *Let $G = \langle V, E\rangle$ be a graph, let $\langle \Sigma, R\rangle$ be a partition pair on V. $\langle \Sigma, R\rangle$ is said* stable *w.r.t. the transition relation of the graph E iff:*

$$\forall \alpha, \beta \in \Sigma(\alpha \rightarrow_\exists \beta \Rightarrow \bigcup\{\delta \mid \delta \in R(\alpha)\} \rightarrow_\forall \bigcup\{\delta \mid \delta \in R(\beta)\}$$

Definition 7 (Coarsest Partition Pair Problem (CPPP)). *Let $G = \langle V, E, \Sigma\rangle$ be a labelled graph, and consider the identity relation I on Σ. The coarsest partition pair problem asks to determine the coarsest partition pair $\langle \Pi, P\rangle \sqsubseteq \langle \Sigma, I\rangle$ stable w.r.t. E.*

Lemma 1 (CPPP as Simulation Problem). *Let $G = \langle V, E, \Sigma\rangle$ be a labelled graph. The coarsest partition pair problem is well defined and admits as unique solution the partition pair $\langle V_{\equiv_S}, S\rangle$, corresponding to the maximum simulation preorder on G consistent w.r.t. Σ.*

Proof. We show that the unique solution to the CPPP is the partition pair $\langle V_{\equiv_S}, S\rangle \sqsubseteq \langle \Sigma, I\rangle$ corresponding to the maximum simulation preorder $\preceq_S$ on $G = \langle V, E\rangle$ consistent w.r.t. Σ. We start by proving that $\langle V_{\equiv_S}, S\rangle$ is stable w.r.t. E. Let $\alpha, \beta \in V_{\equiv_S}$ and assume that $\alpha \rightarrow_\exists \beta$. Then, there exist two nodes $s \in \alpha$, $s' \in \beta$ such that $s \rightarrow s'$. Consider an arbitrary node $p \in \bigcup\{\delta \mid \delta \in S(\alpha)\}$. Since $s \preceq_S p$ and $s \rightarrow s'$, there exists a node p' such that $p \rightarrow p'$ and $s' \preceq_S p'$. Hence, $p' \in \bigcup\{\delta \mid \delta \in S(\beta)\}$. Our arbitrary choice of $p \in \bigcup\{\delta \mid \delta \in S(\alpha)\}$ guarantees that $\bigcup\{\delta \mid \delta \in S(\alpha)\} \rightarrow_\forall \bigcup\{\delta \mid \delta \in S(\beta)\}$, i.e. $\langle V_{\equiv_S}, S\rangle$ is stable w.r.t E.

To conclude our thesis, assume by absurd that there exists a partition pair $\langle \Pi, P\rangle \sqsubseteq \langle \Sigma, I\rangle$ stable w.r.t. E and such that $\neg(\langle \Pi, P\rangle \sqsubseteq \langle V_{\equiv_S}, S\rangle)$. Consider the relation $<_{\langle \Pi,P\rangle} \subseteq V \times V$, where $<_{\langle \Pi,P\rangle} = \{(s, s') \mid ([s]_\Pi, [s']_\Pi) \in P\}$. By our assumption stating that $\neg(\langle \Pi, P\rangle \sqsubseteq \langle V_{\equiv_S}, S\rangle)$, we have that $<_{\langle \Pi,P\rangle} \not\subseteq \preceq_S$. Hence, an absurd follows by proving that $<_{\langle \Pi,P\rangle}$ is a simulation on $G = \langle V, E\rangle$ consistent w.r.t. Σ. In fact, in that case the relation $<_{\langle \Pi,P\rangle} \cup \preceq_S \supset \preceq_S$ would be a simulation relation strictly including the maximum simulation preorder $\preceq_S$. To prove that $<_{\langle \Pi,P\rangle}$ is a simulation on $G = \langle V, E\rangle$ consistent w.r.t. Σ, let $(s, s') \in <_{\langle \Pi,P\rangle}$. By $\langle \Pi, P\rangle \sqsubseteq \langle \Sigma, I\rangle$, we have that $[s]_\Sigma = [s']_\Sigma$. Consider p such that $s \rightarrow p$. Then $[s]_\Pi \rightarrow_\exists [p]_\Pi$. Since $\langle \Pi, P\rangle$ is stable w.r.t. E we have that $s' \in \bigcup\{\delta \mid \delta \in P([s]_\Pi)\}$ has an edge to a node $p' \in \bigcup\{\delta \mid \delta \in P([p]_\Pi)\}$, i.e. to a node p' such that $(p, p') \in <_{\langle \Pi,P\rangle}$. $\square$

3 An Optimal Simulation Algorithm on Acyclic Graphs

In this section, we introduce an optimal simulation algorithm (w.r.t. both time and space) on acyclic graphs. Such a procedure relies on solving the coarsest partition pair problem (i.e. computing the maximum simulation preorder) proceeding by *rank*. The notion of rank, introduced in Definition 8, allows one to perform a preliminary partition in the given labelled graph. This is useful to drive the successive computation, as stated in Lemma 2.

Definition 8. *Let $G = \langle V, E\rangle$ be an acyclic graph, let $v \in V$. The* rank *of the node v is defined as:*

$$rank(v) = \begin{cases} 0 & \text{if } E(v) = \emptyset, \\ \max\{1 + rank(u) \mid (v, u) \in E\} & \text{otherwise.} \end{cases}$$

Example 3. Consider the labelled graph in Figure 1, described in Example 2. In such a graph, node x has rank 2, node y has rank 1 and node z has rank 0.

Lemma 2 (Rank & Simulation). *Let $G = \langle V, E\rangle$ be an acyclic graph, and consider a partition Σ on V. Then:*

$$\langle V_{\equiv_S}, S\rangle \sqsubseteq \langle V_{\equiv_R}, R\rangle$$

where $\langle V_{\equiv_S}, S\rangle$ is the partition pair on V encoding the maximum simulation consistent w.r.t. Σ, and $\langle V_{\equiv_R}, R\rangle$ is the partition pair on V corresponding to the rank-labelling preorder $\preceq_R = \{(u, v) \mid rank(u) \leq rank(v)\}$.

Proof. By absurd, assume that $\neg(\langle V_{\equiv_S}, S\rangle \sqsubseteq \langle V_{\equiv_R}, R\rangle)$. It follows that the maximum simulation preorder $\preceq_S$ is not included in the relation $\{(s,s') \mid ([s]_{\equiv_R}, [s']_{\equiv_R}) \in R\}$, i.e. there exists two nodes $s \in V$, $s' \in V$ such that $s \preceq_S s'$ and $\mathit{rank}(s) > \mathit{rank}(s')$. Let $r = \mathit{rank}(s) > \mathit{rank}(s')$. Since $\mathit{rank}(s) = r$, we can determine a sequence of r nodes $s_1, \dots, s_r$ such that $s \to s_1 \bigwedge_{1\le i<r} s_i \to s_{i+1}$. By definition of simulation, there exists a corresponding sequence of r nodes $s'_1, \dots, s'_r$ such that

$$(s' \to s'_1 \wedge s_1 \preceq_S s'_1) \bigwedge_{1\le i<r} s_i \to s_{i+1} \wedge s_{i+1} \preceq_S s'_{i+1}$$

Such a sequence of nodes witnesses the absurd $\mathit{rank}(s') \geq r$. □

Table 1 shows our rank-based simulation procedure on acyclic graphs. Such an algorithm consists, mainly, of two phases: A preprocessing step in line 1–11 and a main loop in line 12–28.

The preprocessing step performs the rank partitioning of the given labelled graph, which is then explored proceeding by rank (from the lowest to the highest rank), within successive executions of the main loop at line 12. At each iteration i of such a loop (corresponding to rank $i-1$) the transitions targeting nodes having rank $i-1$ are employed to refine the current partition pair, in order to establish the stability property.

Example 4. Consider the labelled graph $G = \langle V, E, \Sigma\rangle$ depicted in Figure 1, illustrated within Example 2. The algorithm SolveCPPP in Table 1 terminates upon the execution of the first loop at line 3, that uses the information given by the rank to refine the initial partition pair $\langle \Sigma, I\rangle$ to the partition pair $\langle \Pi, P\rangle$, where $\Pi = \{\alpha = \{x\}, \alpha_1 = \{y\}, \beta = \{z\}\}$ and $P = I \cup \{(\alpha_1, \alpha)\}$. In fact, in this case the preprocessing provided by the loop at line 3 is sufficient to solve the simulation problem.

4 Correctness and Complexity Results

In this section, we prove the correctness of our simulation algorithm on acyclic graphs, as well as its complexity.

4.1 Correctness

Lemma 3, below, shows that the preprocessing step correctly computes the partition pair induced by the rank labelling of the graph.

Lemma 3. *The loop at line* 3 *in the algorithm* SolveCPPP$(\langle V, E\rangle, \langle \Sigma, I\rangle)$ *terminates computing the partition pair* $\langle \Pi, P\rangle$ *and the variable rankMax, where:*

- *rankMax*$= \max\{\mathit{rank}(v) \mid v \in V\}$,
- $\Pi \sqsubseteq \Sigma$ *is the coarsest partition finer than the rank-labelling partition* $V_{\equiv_R}$,
- $P = \{(\alpha,\beta) \mid \mathit{rank}(\alpha) \leq \mathit{rank}(\beta) \wedge \exists \gamma \in \Sigma(\alpha \subseteq \gamma \wedge \beta \subseteq \gamma)\}$

Proof. This can be easily proved by induction on the number of loop iterations. □

Algorithm 1: SOLVECPPP

```
input  : $G = \langle V, E\rangle$,$\langle \Sigma, I = \{(\alpha,\alpha) \mid \alpha \in \Sigma\}\rangle$
output: $\langle V_{\equiv_S}, S\rangle$: partition pair encoding the maximum simulation preorder $\preceq_S$ on $G$
        consistent w.r.t. $\Sigma$.
1  begin
       /* Initial refinement & rank-ordering of the classes.   */
2      $notRanked := V; rankMax := -1$
3      repeat
4          forall $\alpha \in \Sigma \mid \alpha \subseteq notRanked \wedge \alpha \nsubseteq pre(notRanked)$ do
5              if $\alpha \neq \alpha \setminus pre(notRanked)$ then
6                  $\alpha_1 := \alpha \setminus pre(notRanked); \alpha := \alpha \setminus \alpha_1; rank(\alpha_1) := rankMax + 1;$
                   $\Sigma := \Sigma \cup \{\alpha_1\}; sim := sim \cup \{(\gamma, \alpha_1) \mid (\gamma, \alpha) \in sim\} \cup \{(\alpha_1, \alpha)\}$
7              else
8                  $rank(\alpha) := rankMax + 1$
9
10         $notRanked := pre(notRanked); rankMax := rankMax + 1$
11     until $pre(notRanked) = \emptyset$
       /* Process $\Sigma$ by rank & refine $\langle \Sigma, sim\rangle$ to establish the
          stability prop. $\forall(\alpha,\beta).(\alpha \rightarrow_\exists \beta \Rightarrow sim(\alpha) \rightarrow_\forall sim(\beta))$.      */
12     for $rk = 1$ to $rankMax$ do
13         foreach $\beta \in \Sigma \mid rank(\beta) = rk - 1$ do
14             foreach $\alpha \mid pre(\beta) \cap \alpha \neq \emptyset$ do
15                 if $\alpha \nsubseteq pre(sim(\beta))$ then
16                     $\alpha_1 := \alpha \setminus pre(sim(\beta)); rank(\alpha_1) := rank(\alpha); \alpha := \alpha \setminus \alpha_1;$
                       $\Sigma := \Sigma \cup \alpha_1$
17                     $sim := sim \cup \{(\alpha_1, \delta) \mid (\alpha, \delta) \in sim\}$
18                                 $\cup\{(\delta, \alpha_1) \mid (\delta \neq \alpha, \alpha) \in sim\}$
19                 foreach $\gamma \mid \gamma \nsubseteq pre(sim(\beta)) \wedge \gamma \in sim(\alpha)$ do
20                     $\gamma_1 := \gamma \setminus pre(sim(\beta))$
21                     if $\gamma_1 \neq \gamma$ then
22                         $\gamma := \gamma \setminus \gamma_1; rank(\gamma_1) := rank(\gamma); \Sigma := \Sigma \cup \gamma_1$ $sim :=$
                           $sim \cup \{(\gamma_1, \delta) \mid (\gamma, \delta) \in sim\} \cup \{(\delta, \gamma_1) \mid (\delta \neq \gamma, \gamma) \in sim\}$
23                     $sim := sim \setminus \{(\alpha, \gamma_1)\}$
24
25
26
27
28 end
```

Lemma 4 and Theorem 1 define crucial invariants and prove their validity throughout the execution of the main loop at line 12. Such invariants define the correctness of our simulation algorithm.

Lemma 4. *Consider the overall execution of the for-loop at line* 12 *guarded by the variable* $1 \leq rk \leq rankMax$. *Whenever a class* γ *is involved either in a split or in a refinement of its simulator set* $\bigcup\{\beta \mid (\gamma, \beta) \in sim\}$, *the following statement holds:*

$$\textit{rank}\,(\gamma) \geq rk$$

Proof. Denote by $\langle \Sigma_i, sim_i \rangle$ the partition pair in input to the i-th iteration of the for-loop at line 12. Moreover, if γ is a class processed within the i-th execution of the for-loop at lines 12, let γ_i denote the unique class $\gamma_i \in \Sigma_i$ such that $\gamma_i \supseteq \gamma$.

Given the above notations, consider the i-th execution of the for-loop at lines 12 (for which $rk = i$) and let γ be a class in a partition pair processed within such an iteration. Assume that either $\gamma \subset \gamma_i$, or $\bigcup\{\beta \mid (\gamma, \beta) \in sim\} \neq \bigcup\{\beta_i \mid (\gamma_i, \beta_i) \in sim_i\}$. Then, there exists a pair of classed $\{\alpha, \beta\} \subseteq \Sigma_i$ such that $\textit{rank}\,(\beta) = i - 1$, $\alpha \rightarrow_\exists \beta$ and $\gamma_i \in sim_i(\alpha)$. $\textit{rank}\,(\beta) = i - 1 \wedge \alpha \rightarrow_\exists \beta$ implies $\textit{rank}\,(\alpha) \geq i$. Lemma 3 allows then to conclude that $\textit{rank}\,(\gamma) = \textit{rank}\,(\gamma_i) \geq i$. □

Theorem 1. *The following invariants hold at the beginning of each iteration of the for-loop at line* 12, *within the the algorithm* SolveCPPP$(\langle V, E \rangle, \langle \Sigma_0, I \rangle)$.

1. *For each node* $v \in V$ *such that rank* $(v) < rk$:

$$[v]_{\equiv_S} = \alpha \in \Sigma \wedge \bigcup\{u \mid v \preceq_S u\} = \bigcup\{\beta \mid (\alpha, \beta) \in sim\}$$

2. *For each node* $v \in V$ *such that rank* $(v) \geq rk$:

$$[v]_{\equiv_S} \subseteq \alpha \in \Sigma \wedge \bigcup\{u \mid v \preceq_S u\} \subseteq \bigcup\{\beta \mid (\alpha, \beta) \in sim\}$$

3. *For each pair of classes* $\alpha, \beta \in \Sigma$:

$$(\alpha \rightarrow_\exists \beta \wedge \textit{rank}\,(\beta) < rk - 1) \Rightarrow \bigcup\{\gamma \mid (\alpha, \gamma) \in sim\} \rightarrow_\forall \bigcup\{\gamma \mid (\beta, \gamma) \in sim\}$$

Proof. By induction on the number of iterations of the for-loop at line 11.
Base $(rk = 1)$. The first two items in our statement follow directly from Lemma 2 and Lemma 3, while the third item holds trivially since no class $\beta \in \Sigma$ has a rank strictly lower than $rk - 1 = 0$.
Inductive Step $(1 < rk)$. Given $i \geq 1$, denote by $\langle \Sigma_i, sim_i \rangle$ the partition pair in input to the i-th execution of the for-loop at line 12. Given $\gamma \in \Sigma_{i>1}$, denote by γ_{i-1} the only class in Σ_{i-1} such that $\gamma_{i-1} \supseteq \gamma$.

1. In order to prove our inductive step for item (3), consider $\langle \Sigma_{i=rk>1}, sim_{i=rk>1} \rangle$ and let $\alpha \in \Sigma_i$ such that $\alpha \rightarrow \beta \wedge \textit{rank}\,(\beta) < i - 1$. By $\textit{rank}\,(\beta) < i - 1$ and Lemma 4 we have $\beta_{i-1} = \beta$ and $\bigcup\{\delta \mid (\beta, \delta) \in sim_i\} = \bigcup\{\delta \mid (\beta_{i-1}, \delta) \in sim_{i-1}\} = sim\beta$. Hence, if $\textit{rank}\,(\beta) < (i - 1) - 1$ we can conclude our thesis exploiting the inductive hypothesis for which $\bigcup\{\delta \mid (\alpha_{i-1}, \delta) \in sim_{i-1}\} \rightarrow_\forall$

$sim\beta$. Otherwise, assume *rank* $(\beta) = i - 1$ and suppose by contradiction that $\neg(\bigcup\{\delta \mid (\alpha, \delta) \in sim_i\} \rightarrow_\forall sim\beta)$. Let $\alpha^* \supset \alpha$ be the superclass of α at the moment in which β gets selected at line 13 with $rk = i - 1$. Within the execution of the most internal foreach-loop at line 14, if α^* contains some state that does not reach $sim\beta$, then α^* gets split into the two subclasses: $\alpha_1^* := \alpha^* \setminus sim\beta$ and $\alpha^* := \alpha^* \setminus \alpha_1$. By $\alpha \rightarrow_\exists \beta \subseteq sim\beta$, we have that the statement $\alpha \subseteq \alpha^*$ is true both before and after such a split. Moreover, the loop at line 19 processes each class $\gamma \in sim_{i-1}^*(\alpha^* \supseteq \alpha)$. If γ contains some state that does not reach $sim\beta$, then γ gets split into the two subclasses $\gamma_1 := \gamma \setminus sim\beta$ and $\gamma := \gamma \setminus \alpha_1$, and γ_1 is removed from the simulators of $\alpha^* \supseteq \alpha$ (line 23). Hence, once β have been considered at line 13 within the $i - 1$-th execution of the for-loop at line 12, we have that each node belonging to a class simulating α^* has a successor in $sim\beta$. Each subsequent refinement of α^* or its set of simulators will maintain this property, and thus we get to the contradiction of our assumption $\neg(\bigcup\{\delta \mid (\alpha, \delta) \in sim_i\} \rightarrow_\forall sim\beta$

2. We now proceed proving the inductive step for item (1). Let $i = rk > 1$ and consider the $\langle \Sigma_i, sim_i \rangle$. Let $v \in V$ such that *rank* $(v) < rk - 1$. Then, item (1) holds by Lemma 4 and by inductive hypothesis. Let v such that *rank* $(v) = rk - 1$. Then, by inductive hypothesis on item (2) we have:

$$[v]_{\equiv_S} \subseteq \alpha \in \Sigma \wedge \bigcup\{u \mid v \preceq_S u\} \subseteq \bigcup\{\beta \mid (\alpha, \beta) \in sim\} \tag{1}$$

Moreover, the inductive step already proved on item (3) ensures that:

$$(\alpha \rightarrow_\exists \beta \wedge \textit{rank}\,(\beta) < rk - 1) \Rightarrow \bigcup\{\gamma \mid (\alpha, \gamma) \in sim\} \rightarrow_\forall \bigcup\{\gamma \mid (\beta, \gamma) \in sim\} \tag{2}$$

Since *rank* $(\alpha) = rk - 1$, any class reached by α has rank strictly lower than $rk - 1$. Hence, equations 1 and 2 guarantee that:

$$[v]_{\equiv_S} = \alpha \in \Sigma \wedge \bigcup\{u \mid v \preceq_S u\} = \bigcup\{\beta \mid (\alpha, \beta) \in sim\}$$

completing our inductive step for item (1).

3. Let $i = rk > 1$ and consider v such that $rank(v) \geq rk$. If $rank(v) > rk$, than $[v]_{\Sigma_i} = [v]_{\Sigma_{i+1}} \subseteq \alpha$ by inductive hypothesis. If $rank(v) = rk$, there are two cases to consider. In the first case, $[v]_{\Sigma_i} = [v]_{\Sigma_{i+1}}$ (i.e. the class of v gets not split within the i-th iteration of the loop at line 13) and we are done. In the second case, $[v]_{\Sigma_i}$ gets split into α, α_1 within the i-th execution of the loop at line 13. By contradiction, assume there exist two states $u \in \alpha, u' \in \alpha_1$ such that $u \equiv_S u'$. By definition of α, α_1, u has a successor into a class β of rank $r < rk$, and u' has no successor in any class $\beta' \in sim(\beta)$. By inductive hypothesis, this implies that there exists z such that $(u, z) \in E$ and u' has no successor z' such that $z \leq_S z'$, contradicting our hypothesis $u \equiv_S u'$.

□

4.2 Complexity

We finally establish the complexity of our simulation algorithm on acyclic graphs. In particular, Theorem 2 shows that our procedure uses $\mathcal{O}(|E||V_{\equiv_S}|)$ time and requires

$O(|V_{\equiv_S}|^2 + |V|\log(|V_{\equiv_S}|))$ bits to compute a simulation preorder on a given acyclic labelled graph $G = \langle V, E, \Sigma\rangle$. Therefore, it has optimal performances w.r.t. both time and space on acyclic graphs, outperforming [17, 18, 9].

Theorem 2. *The algorithm* SolveCPPP$(G = \langle V, E\rangle, \langle \Sigma, I\rangle)$ *performs* $O(|V_{\equiv_S}||E|)$ *steps and uses* $O(|V_{\equiv_S}|^2 + |V|\log(|V_{\equiv_S}|))$ *bits to compute the solution to the coarsest partition pair problem* $\langle V_{\equiv_S}, S\rangle$.

Proof. Let $r = \max\{\mathit{rank}(v) \mid v \in V\}$. The cost of the while-loop at line 3 is $O(r * |E|) = O(|V_{\equiv_S}||E|)$.

The cost of the loop at line 12, excluded the execution of the innermost if-statement at line 21, is:

$$O(\Sigma_{1=1}^{r}\Sigma_{\beta\in V_{\equiv_S}, \mathit{rank}(\beta)=i-1}(|pre(\beta)| * |\Sigma_{i+1}| + |pre(\bigcup\{\delta \mid (\beta,\delta)\in sim\}|) =$$

$$= O(|V_{\equiv_S}||E|)$$

In fact, consider a class β such that $\mathit{rank}(\beta) = i - 1$. It is possible to distinguish with marks the classes that reach (resp. do not reach/reach with all their nodes) the set $\bigcup\{\delta \mid (\beta,\delta)\in sim\}$ at the cost $O(pre(\bigcup\{\delta \mid (\beta,\delta)\in sim\})$. Moreover, the same cost allows one to appropriately mark each node in $pre(\bigcup\{\delta \mid (\beta,\delta)\in sim\}$. Then, fixed $\beta, \mathit{rank}(\beta) = i-1$, the cost of executing lines 14–24 without considering the innermost if-statement, is $O(|pre(\bigcup\{\delta \mid (\beta,\delta)\in sim\}| + \Sigma_{\alpha\in pre(\beta)}|sim(\alpha)|)$.

The innermost if-statement at lines 21–23 is executed only upon the creation of a new class γ_1 and cost globally $O(|V_{\equiv_S}||E|)$. In fact, each execution of lines 21–23 for the creation of the new classes $\gamma_1, \gamma\setminus\gamma_1$ from γ, requires only to scan the nodes in γ and the classes in $sim(\gamma), sim^{-1}(\gamma)$.

As far as space complexity is concerned we refer to bit complexity without considering the space required by the graph G, since it is never modified (see, e.g., [15, 2, 3]). In particular, Σ is stored through an array of length $|V|$ associating to each node its class. Hence, it requires $O(|V|\log(|V_{\equiv_S}|))$ bits. $notRanked$ is a $|V|$ array of bits, labeling with 1 the nodes which do not have a rank. $rank$ is stored in a $|V|$ array of lists, where the ith list keeps the classes at rank i. Hence it requires $O(|V| + |V_{\equiv_S}|\log(|V_{\equiv_S}|))$ bits. Finally, the relation sim is stored in a bit matrix whose size grows up to $O(|V_{\equiv_S}|^2)$ bits. □

5 Conclusion

We presented an algorithm for computing the maximum simulation quotient and the simulation preorder on acyclic graphs. Our algorithm is based on a characterization of the simulation problem as coarsest partition pair problem and on the notion of rank. The notion of rank is a standard one in well-founded set theory (see, e.g., [19]) and has been exploited in [7, 8] to define an algorithm for computing the maximum bisimulation. The algorithm presented in [7, 8] has a linear time complexity in the acyclic case, while in the general case it allows one to focus the computation on subgraphs of the given graph. The algorithm we presented here has optimal space/time performances on acyclic graphs.

On the one hand, its generalization to the general case is still an open problem. Unfortunately, when moving from acyclic to cyclic graphs the notion of simulation is not "compositional", since loops allow to simulate paths of arbitrary length.

On the other hand, another open problem concerns the possibility of developing a linear time simulation algorithm for the acyclic case. Again, there seems to be an intrinsic higher complexity in the notion of simulation w.r.t. bisimulation which does not allow to reach the linear time complexity. In particular, while the notion of bisimulation on acyclic graphs corresponds to equality on well-founded sets, simulation in terms of set theory is a sort of recursive inclusion.

References

1. Bard Bloom and Robert Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Science of Computer Programming*, 24(3):189–220, June 1995.
2. D. Bustan and O. Grumberg. Simulation based minimization. In D.A. McAllester, editor, *Proc. 17th Int'l Conference on Automated Deduction (CADE'00)*, volume 1831 of *LNCS*, pages 255–270. Springer, 2000.
3. D. Bustan and O. Grumberg. Simulation based minimization. *ACM Transactions on Computational Logic (TOCL)*, 4:181–206, 2003.
4. E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logics. *Logic of Programs*, pages 52–71, 1981.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. Elsevier/MIT press, 2001.
6. R. Cleaveland and L. Tan. Simulation revisited. In T. Margaria and W. Yi, editors, *Proc. 7th Int'l Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *LNCS*, pages 480–495. Springer, 2001.
7. A. Dovier, C. Piazza, and A. Policriti. A fast bisimulation algorithm. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 79–90. Springer, 2001.
8. A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Science*, 311:221–256, 2004.
9. R. Gentilini, C. Piazza, and A. Policriti. From bisimulation to simulation: Coarsest partition problems. *J. Autom. Reasoning*, 31(1):73–103, 2003.
10. M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 453–462. IEEE Computer Society Press, 1995.
11. K. Laiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1):11–44, 1995.
12. R. Milner. A calculus of communicating systems. In G. Goos and J. Hartmanis, editors, *Lecture Notes on Computer Science*, volume 92. Springer, 1980.
13. O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and systems*, 16(3):843–871, May 1994.
14. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
15. C. H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Inc., 1994.
16. D. Park. Concurrency on automata and infinite sequences. *Theoretical Computer Science*, pages 167–183, 1981.

17. F. Ranzato and F. Topparo. A new efficient simulation equivalence algorithm. In *Proceedings of Logics in Computer Science (LICS'07)*, pages 171–180, 2007.
18. F. Ranzato and F. Topparo. Saving space in a time efficient simulation algorithm. In *Proceedings of Int. Conference on Application of Concurrency to System design (ACSD'09)*, pages 60–69, 2009.
19. J. E. Rubin. *Set Theory for the Mathematician*. New York: Holden-Day, 1967.
20. R. van Glabbeek and B. Ploeger. Correcting a space-efficien simulation algorithm. In *Proceedings of Int. Conference on Computer Aided Verification (CAV'08)*, pages 517–529, 2008.