Correcting a space-efficient simulation algorithm

Correcting a Space-Efficient Simulation Algorithm

Rob van Glabbeek[1,2] Bas Ploeger[3*]
rvg@cs.stanford.edu s.c.w.ploeger@tue.nl

[1] National ICT Australia, Locked Bag 6016, Sydney, NSW1466, Australia

[2] School of Computer Science and Engineering, The University of New South Wales
Sydney, NSW 2052, Australia

[3] Department of Mathematics and Computer Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract

Although there are many efficient algorithms for calculating the simulation preorder on finite Kripke structures, only two have been proposed of which the space complexity is of the same order as the size of the output of the algorithm. Of these, the one with the best time complexity exploits the representation of the simulation problem as a generalised coarsest partition problem. It is based on a fixed-point operator for obtaining a generalised coarsest partition as the limit of a sequence of partition pairs. We show that this fixed-point theory is flawed, and that the algorithm is incorrect. Although we do not see how the fixed-point operator can be repaired, we correct the algorithm without affecting its space and time complexity.

1 Introduction

The *simulation preorder* [16] is a behavioural refinement relation on concurrent systems, represented as Kripke structures or labelled transition systems, that plays a crucial rôle in compositional verification and model checking. As shown in [5] and [15], respectively, the simulation preorder preserves the existential and universal fragments of CTL* [6], as well as the modal μ-calculus [13]. This makes it possible to combat the state explosion problem in model checking by minimising the state space of a given system modulo simulation equivalence before checking the validity of relevant properties within that fragment. Given that the simulation preorder is a precongruence for parallel composition [10], components in parallel compositions can even be minimised individually.

Simulation equivalence is also used directly in equivalence checking [14] of finite-state processes. Often deciding the simulation preorder between processes is the most appropriate method of showing that two systems are related by another preorder, that may be appropriate for the task at hand. In applications where deadlock behaviour

*This author is partially supported by the Netherlands Organisation for Scientific Research (NWO) under VoLTS grant number 612.065.410.

plays a crucial rôle, the *ready simulation preorder* [1] is widely regarded to be an appropriate behavioural refinement relation for matching an implementation with a specification. Via a straightforward reduction (the computation of the initial partition ER_1 in [2]), finding a ready simulation between two processes is as hard as finding a plain simulation. In applications where deadlock behaviour plays no rôle, *trace inclusion* is often proposed as an appropriate refinement relation. However, deciding trace inclusion on finite-state processes is PSPACE-hard [18], and as the simulation preorder is the coarsest preorder included in trace inclusion that is known to be decidable in polynomial time [2, 3, 8, 11, 17, 19], establishing a simulation between two processes is a favourite way of showing that they are related by trace inclusion.

In many crucial applications, space rather than time becomes the bottleneck as the input graph grows [4, 7, 8, 12]. Hence, simulation algorithms with minimal space complexity are of particular interest. These are the ones by Bustan and Grumberg [3] and by Gentilini, Piazza and Policriti [8]. For an input graph with N states, T transitions and S simulation equivalence classes, the space complexity of both algorithms is $\mathcal{O}(S^2 + N \log S)$. This can be considered minimal: $\mathcal{O}(S^2)$ space is needed for storing the simulation preorder as a partial order on simulation equivalence classes and $\mathcal{O}(N \log S)$ space is needed to store for every state, the equivalence class to which it belongs. Of these algorithms, the one by Gentilini *et al.* has a better time complexity: $\mathcal{O}(S^2 T)$. A more time-efficient algorithm is the one by Ranzato and Tapparo [17], but it is less space efficient.

The approach of Gentilini *et al.* represents the simulation problem as a generalised coarsest partition problem (GCPP). According to the authors, this problem can be solved by approximating the greatest fixed point of a decreasing operator on partition pairs that they define in their paper. They give a partitioning algorithm to compute this fixed point for any legal input. We recite this definition and a part of the algorithm in Section 3. In Section 4 we show that the operator is flawed because it is not uniquely defined for all partition pairs. We give an instance of the GCPP for which repeated application of the operator does not lead to a unique fixed point. We also show that on this example the partitioning algorithm irrevocably allocates two simulation-equivalent states to different simulation-equivalence classes, and subsequently deadlocks.

In Section 5 we define a simple, yet inefficient fixed-point operator for which we prove correctness. This operator is not meant to be an improvement over the original one, but merely serves as an expedient for establishing correctness of the algorithm that we present in Section 6. This algorithm is obtained from that of Gentilini *et al.* by means of a few simple corrections; consequently, it has the same time and space complexities as the original partitioning algorithm. Yet its correctness proof requires entirely new techniques and is surprisingly non-trivial. We also show that no fixed-point operator can be defined that captures the behaviour of this algorithm.

2 Preliminaries

Partitions and relations. For any set S, a *partition* over S is a set $\Sigma \subseteq \mathcal{P}(S)$ such that $\bigcup \Sigma = S$ and $\forall \alpha \in \Sigma \,.\, \alpha \neq \emptyset \wedge \forall \beta \in \Sigma \,.\, \alpha \neq \beta \Rightarrow \alpha \cap \beta = \emptyset$. For any $s \in S$ we denote by $[s]_\Sigma$ the block $\alpha \in \Sigma$ such that $s \in \alpha$. Given two partitions Σ and Π we say Π *is finer than* Σ iff for every $\alpha \in \Pi$ there exists an $\alpha' \in \Sigma$ such that $\alpha \subseteq \alpha'$. For any set S, we denote by $\mathcal{I}(S)$ the *identity relation* over S, *i.e.* $\mathcal{I}(S) = \{(s,s) \mid s \in S\}$. For any relation P, we denote by P^+ the *transitive closure* of P.

Graphs. A *(directed) graph* is a tuple $(N, \rightarrow)$ where N is a finite set of nodes and $\rightarrow \subseteq N \times N$ is a set of directed transitions between those nodes. A *labelled graph* is a tuple $(N, \rightarrow, \Sigma)$ where $(N, \rightarrow)$ is a graph and Σ is a partition over N. For a graph $(N, \rightarrow)$, $a \in N$ and $\beta \subseteq N$, we write $a \rightarrow \beta$ if $\exists b \in \beta \,.\, a \rightarrow b$. Moreover, we define the relations $\rightarrow_\exists$ and $\rightarrow_\forall$ over $\mathcal{P}(N)$ as follows, for any $\alpha, \beta \subseteq N$:

$$\alpha \rightarrow_\exists \beta \Leftrightarrow \exists a \in \alpha \,.\, a \rightarrow \beta \qquad\qquad \alpha \rightarrow_\forall \beta \Leftrightarrow \forall a \in \alpha \,.\, a \rightarrow \beta.$$

Simulations. For any labelled graph $(N, \rightarrow, \Sigma)$ a relation $R \subseteq N \times N$ is a *simulation* iff for any $a, b \in N$, $(a, b) \in R$ implies:

- $[a]_\Sigma = [b]_\Sigma$ and
- $\forall c \in N \,.\, a \rightarrow c \Rightarrow \exists d \in N \,.\, b \rightarrow d \wedge (c, d) \in R$.

We say that a is *simulated by* b, denoted $a \subsetneq_{\rightarrow} b$, iff there exists a simulation R such that $(a, b) \in R$. It is well known and easy to check that $\subsetneq_{\rightarrow}$ is a preorder, *i.e.* a reflexive and transitive relation, on N, and moreover the largest simulation. We say that a and b are *simulation equivalent*, denoted $a \rightleftarrows b$, iff $a \subsetneq_{\rightarrow} b$ and $b \subsetneq_{\rightarrow} a$.

The simulation problem. Given a labelled graph $G = (N, \rightarrow, \Sigma)$, the *simulation problem* over G consists in finding the simulation preorder $\subsetneq_{\rightarrow}$ on G.

A variant of the simulation problem asks, given a labelled graph $(N, \rightarrow, \Sigma)$ and two nodes $a, b \in N$, whether $a \subsetneq_{\rightarrow} b$. In general, no methods to solve this problem are known that are more efficient than computing the entire relation $\subsetneq_{\rightarrow} \subseteq N \times N$ and looking up whether $(a, b) \in \subsetneq_{\rightarrow}$. Another variant of the simulation problem merely asks to find the simulation equivalence relation $\rightleftarrows$ rather than the preorder $\subsetneq_{\rightarrow}$. Again, no methods to solve that problem are known that do not amount to finding $\subsetneq_{\rightarrow}$ as well.

Typically, the simulation problem arises in the context of *Kripke structures* or *labelled transition systems*. It is trivial to encode a Kripke structure as a labelled graph in such a way that the simulation preorder on the Kripke structure agrees with the one on its labelled graph representation. Likewise, it is not hard to reduce the simulation problem for labelled transition systems to that for labelled graphs. Alternatively one can enrich the theory in a straightforward way to deal with transition labels as well, so that it is applicable to labelled transition systems directly.

The generalised coarsest partition problem. Given a graph $G = (N, \rightarrow)$, a *partition pair* over G is a pair $\langle \Sigma, P \rangle$ where Σ is a partition over N and $P \subseteq \Sigma \times \Sigma$ is a reflexive, acyclic relation over Σ. A partition pair $\langle \Sigma, P \rangle$ is called *transitive* if P is transitive, and hence a partial order. Given a partition Σ, a partition Π finer than Σ, and a relation P over Σ, we denote by $P(\Pi)$ the *induced relation* of P on Π:

$$P(\Pi) = \{(\alpha, \beta) \in \Pi \times \Pi \mid \exists (\alpha', \beta') \in P \,.\, \alpha \subseteq \alpha' \wedge \beta \subseteq \beta'\}.$$

We define a *partial order* $\leq$ on partition pairs by writing, for any partition pairs $\langle \Sigma, P \rangle$ and $\langle \Pi, Q \rangle$: $\langle \Pi, Q \rangle \leq \langle \Sigma, P \rangle$ iff Π is finer than Σ and $Q \subseteq P(\Pi)$. Given a graph $G = (N, \rightarrow)$, we say a partition pair $\langle \Sigma, P \rangle$ over G is *stable with respect to* $\rightarrow$ [8] iff:

$$\forall \alpha, \beta, \gamma \in \Sigma \,.\, ((\alpha, \beta) \in P \wedge \alpha \rightarrow_\exists \gamma) \Rightarrow \exists \delta \in \Sigma \,.\, (\gamma, \delta) \in P \wedge \beta \rightarrow_\forall \delta.$$

Given a graph $G = (N, \rightarrow)$ and a partition pair $\langle \Sigma, P \rangle$ over G, the *generalised coarsest partition problem* (GCPP) [8] consists in finding a $\leq$-maximal partition pair $\langle \Xi, \preceq \rangle$ such that $\langle \Xi, \preceq \rangle \leq \langle \Sigma, P^+ \rangle$ and $\langle \Xi, \preceq \rangle$ is stable with respect to $\rightarrow$.

The simulation problem as a GCPP. Let $G = (N, \rightarrow, \Sigma)$ be a labelled graph. Any preorder $\sqsubseteq$ on N can be represented as a partition pair $\mathsf{PP}(\sqsubseteq) := \langle \Pi, \preceq \rangle$, as follows: Π is the set of equivalence classes of N w.r.t. the equivalence relation $\equiv \; := \; \sqsubseteq \cap \sqsubseteq^{-1}$ induced by $\sqsubseteq$, and $\preceq$ is given by $[a]_\Pi \preceq [b]_\Pi$ iff $a \sqsubseteq b$. Note that $\preceq$ is a partial order. Moreover, if $\sqsubseteq$ is a simulation then $\mathsf{PP}(\sqsubseteq)$ is stable w.r.t. $\rightarrow$ and $\mathsf{PP}(\sqsubseteq) \leq \langle \Sigma, \mathcal{I}(\Sigma) \rangle$.[1]

Any partition pair $\langle \Pi, Q \rangle$ over the graph $(N, \rightarrow)$ can be represented as a relation $R_{\langle \Pi, Q \rangle} \subseteq N \times N$ as follows: $(a, b) \in R_{\langle \Pi, Q \rangle}$ iff $\exists (\alpha, \beta) \in Q \;.\; a \in \alpha \wedge b \in \beta$. Note that if $\langle \Pi, Q \rangle$ is stable w.r.t. $\rightarrow$ and $\langle \Pi, Q \rangle \leq \langle \Sigma, \mathcal{I}(\Sigma) \rangle$ then $R_{\langle \Pi, Q \rangle}$ is a simulation. Moreover, $\langle \Pi, Q \rangle \leq \langle \Pi', Q' \rangle$ iff $R_{\langle \Pi, Q \rangle} \subseteq R_{\langle \Pi', Q' \rangle}$. Also note that $R_{\mathsf{PP}(\sqsubseteq)} = \; \sqsubseteq$.

Hence $\mathsf{PP}(\subseteq_{\rightarrow})$ is the solution of the GCPP on $(N, \rightarrow)$ and $\langle \Sigma, \mathcal{I}(\Sigma) \rangle$. In particular, the GCPP, when applied to partition pairs of the form $\langle \Sigma, \mathcal{I}(\Sigma) \rangle$ (plain partitions), always has a unique solution $\langle \Xi, \preceq \rangle$, in which moreover $\preceq$ is always a partial order.[2]

3 The GCPP Solution of Gentilini, Piazza and Policriti

To solve the GCPP, Gentilini, Piazza and Policriti [8] introduce the following operator:

Definition 4.11 in [8] (Operator σ). Let $G = (N, \rightarrow)$ and $\langle \Sigma, P \rangle$ be a partition pair over G. The partition pair $\langle \Pi, Q \rangle = \sigma(\langle \Sigma, P \rangle)$ is defined as follows:

(1σ) Π is the coarsest partition finer than Σ such that

(a) $\forall \alpha \in \Pi \, \forall \gamma \in \Sigma (\alpha \rightarrow_\exists \gamma \;\Rightarrow\; \exists \delta \in \Sigma ((\gamma, \delta) \in P \wedge \alpha \rightarrow_\forall \delta))$;

(2σ) Q is maximal such that $Q \subseteq P(\Pi)$ and if $(\alpha, \beta) \in Q$, then

(b) $\forall \gamma \in \Sigma (\alpha \rightarrow_\forall \gamma \;\Rightarrow\; \exists \gamma' \in \Sigma ((\gamma, \gamma') \in P \wedge \beta \rightarrow_\exists \gamma'))$ and

(c) $\forall \gamma \in \Pi (\alpha \rightarrow_\forall \gamma \;\Rightarrow\; \exists \gamma' \in \Pi ((\gamma, \gamma') \in Q \wedge \beta \rightarrow_\exists \gamma'))$.

They argue that applying σ iteratively on an initial partition pair $\langle \Sigma_0, P_0 \rangle$ yields a sequence of partition pairs $\langle \Sigma_i, P_i \rangle_{i \geq 0}$ with $\langle \Sigma_{i+1}, P_{i+1} \rangle = \sigma(\langle \Sigma_i, P_i \rangle)$. By construction, this sequence is decreasing, in the sense that $\langle \Sigma_{i+1}, P_{i+1} \rangle \leq \langle \Sigma_i, P_i \rangle$. Hence it will reach a fixed point $\langle \Sigma_k, P_k \rangle = \sigma(\langle \Sigma_k, P_k \rangle)$. This is the solution to the GCPP.

Algorithm 1 The partitioning algorithm of [8]: $\mathrm{PA}_{\mathrm{GPP}}((N, \rightarrow), \langle \Sigma, P \rangle)$

```
change := ⊤; i := 0; Σ_0 := Σ; P_0 := P;
while change do
    change := ⊥;
    Σ_{i+1} := REFINE_GPP(Σ_i, P_i, change);
    P_{i+1} := UPDATE_GPP(Σ_i, P_i, Σ_{i+1});
    i := i + 1;
end while
```

Applying this, they give a partitioning algorithm to solve the GCPP. We have included it here as Algorithm 1 and call it $\mathrm{PA}_{\mathrm{GPP}}$. It takes as input a graph $(N, \rightarrow)$ and a transitive partition pair $\langle \Sigma, P \rangle$ and repeatedly calls the following functions to compute σ until a fixed point is reached: $\textsc{Refine}_{\mathrm{GPP}}$ which computes the partition Π

[1] The proof of the stability claim proceeds similarly to footnote 3 in the proof of Proposition 6.

[2] The same reasoning extends to the GCPP applied to any partition pairs, but this requires considering simulations on structures of the form $(N, \rightarrow, \Sigma, \preceq)$ with $(N, \rightarrow, \Sigma)$ a labelled graph, and $\preceq$ a partial order on Σ; the first clause in the definition of simulation then becomes $[a]_\Sigma \preceq [b]_\Sigma$.

Algorithm 2 The refine function of [8]: $\textsc{Refine}_{\text{GPP}}(\Sigma_i, P_i, \mathit{change})$

1: $\Sigma_{i+1} := \Sigma_i$;
2: **for all** $\alpha \in \Sigma_{i+1}$ **do** $\mathit{Stable}(\alpha) := \emptyset$; **end for**
3: **for all** $\gamma \in \Sigma_i$ **do** $\mathit{Row}(\gamma) := \{\gamma' \mid (\gamma, \gamma') \in P_i\}$; **end for**
4: Let Sort be a reverse topological sorting of Σ_i w.r.t. P_i;
5: **while** $\mathit{Sort} \neq \emptyset$ **do**
6:     $\gamma := \mathit{dequeue}(\mathit{Sort})$;
7:     $A := \emptyset$;
8:     **for all** $\alpha \in \Sigma_{i+1}$, $\alpha \rightarrow_\exists \gamma$, $\mathit{Stable}(\alpha) \cap \mathit{Row}(\gamma) = \emptyset$ **do**
9:         $\alpha_1 := \alpha \cap \rightarrow^{-1}(\gamma)$;
10:         $\alpha_2 := \alpha \setminus \alpha_1$;
11:         **if** $\alpha_2 \neq \emptyset$ **then** $\mathit{change} := \top$; **end if**
12:         $\Sigma_{i+1} := \Sigma_{i+1} \setminus \{\alpha\}$;
13:         $A := A \cup \{\alpha_1, \alpha_2\}$;
14:         $\mathit{Stable}(\alpha_1) := \mathit{Stable}(\alpha) \cup \{\gamma\}$;
15:         $\mathit{Stable}(\alpha_2) := \mathit{Stable}(\alpha)$;
16:     **end for**
17:     $\Sigma_{i+1} := \Sigma_{i+1} \cup A$;
18:     $\mathit{Sort} := \mathit{Sort} \setminus \{\gamma\}$;
19: **end while**
20: **return** Σ_{i+1};

of (1σ) and $\textsc{Update}_{\text{GPP}}$ which computes the relation Q of (2σ). The boolean variable change is set to $\top$ by $\textsc{Refine}_{\text{GPP}}$ iff its output partition differs from its input partition. We have included the $\textsc{Refine}_{\text{GPP}}$ function as Algorithm 2. In line 4 of this algorithm, a "reverse topological sorting of Σ_i w.r.t. P_i" indicates an ordered listing of the elements of Σ_i such that if $(\gamma, \delta) \in P_i$ then δ occurs prior to γ.

4 Incorrectness of the Fixed-Point Operator

Following the definition of σ, the authors claim that for any partition pair $\langle \Sigma, P \rangle$, if $\langle \Pi, Q \rangle = \sigma(\langle \Sigma, P \rangle)$ then Q is acyclic. We give an example that counters this claim.

Counterexample 1. *Consider the graph in Figure 1(a) and the partition pair* $\langle \Sigma, P \rangle$ *with* $\Sigma = \{\alpha, \beta, \gamma, \delta\}$ *as depicted and* $P = \mathcal{I}(\Sigma) \cup \{(\beta, \delta), (\delta, \gamma)\}$*. Let* $\langle \Pi, Q \rangle =$

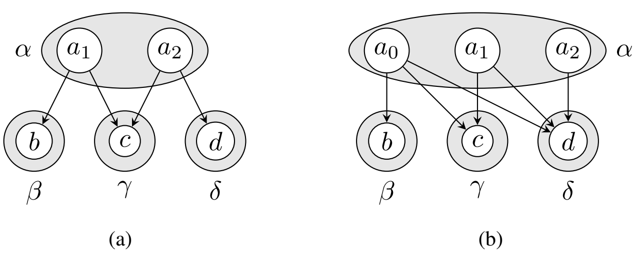


Figure 1: Counterexamples for (a) acyclicity of Q and (b) well-definedness of σ.

$\sigma(\langle \Sigma, P\rangle)$, then

$$\Pi = \{\alpha_1, \alpha_2, \beta, \gamma, \delta\} \qquad Q = \mathcal{I}(\Pi) \cup \{(\alpha_1, \alpha_2), (\alpha_2, \alpha_1), (\beta, \delta), (\delta, \gamma)\}$$

where $\alpha_1 = \{a_1\}$ and $\alpha_2 = \{a_2\}$. Q is not acyclic, which counters the claim. □

This counterexample shows that applying σ to a given partition pair does not necessarily yield another partition pair. After all, for that the resulting relation has to be acyclic.

However, a more fundamental theorem that the authors claim to have proven, turns out not to hold. Theorem 4.13 states that for every partition pair $\langle \Sigma, P\rangle$ there exists a unique $\leq$-maximal partition pair $\langle \Pi, Q\rangle \leq \langle \Sigma, P\rangle$ satisfying conditions (a), (b) and (c) of Definition 4.11, *i.e.* the σ operator is well-defined, and a function. This theorem is countered by the following example.

Counterexample 2. *Consider the graph in Figure 1(b) and the partition pair $\langle \Sigma, P\rangle$ with $\Sigma = \{\alpha, \beta, \gamma, \delta\}$ as depicted and $P = \mathcal{I}(\Sigma) \cup \{(\beta, \gamma), (\gamma, \delta)\}$. Let $\langle \Pi, Q\rangle$ and $\langle \Pi', Q'\rangle$ be partition pairs such that:*

$$\Pi = \{\alpha_0, \alpha_1, \beta, \gamma, \delta\} \qquad Q = \mathcal{I}(\Pi) \cup \{(\alpha_0, \alpha_1), (\alpha_1, \alpha_0), (\beta, \gamma), (\gamma, \delta)\}$$
$$\Pi' = \{\alpha'_0, \alpha'_1, \beta, \gamma, \delta\} \qquad Q' = \mathcal{I}(\Pi') \cup \{(\alpha'_0, \alpha'_1), (\alpha'_1, \alpha'_0), (\beta, \gamma), (\gamma, \delta)\}$$

where $\alpha_0 = \{a_0, a_1\}$, $\alpha_1 = \{a_2\}$, $\alpha'_0 = \{a_0\}$ and $\alpha'_1 = \{a_1, a_2\}$. Both $\langle \Pi, Q\rangle$ and $\langle \Pi', Q'\rangle$ satisfy conditions (a)*,* (b) *and* (c) *of Definition 4.11, but neither is the $\leq$-largest. The only partition pair greater than both $\langle \Pi, Q\rangle$ and $\langle \Pi', Q'\rangle$ and at most as large as $\langle \Sigma, P\rangle$, is $\langle \Sigma, P\rangle$ itself, but $\langle \Sigma, P\rangle$ does not satisfy* (a)*. Hence, this example counters Theorem 4.13 of [8] and shows that σ is not well-defined.* □

Following Theorem 4.13, the authors present their main fixed-point theorem which states that the solution of the GCPP over a graph G and partition pair $\langle \Sigma, P\rangle$ can be computed by applying σ to $\langle \Sigma, P\rangle$ finitely many times until a fixed point is reached (Theorem 4.14). In this theorem, the authors demand that P be transitive. One might be inclined to think that Counterexample 2 does not affect this theorem, as we used a non-transitive P. We now show that this is not the case: the main theorem indeed loses its meaning due to our counterexample for Theorem 4.13. To do so, we first give an example in which the application of σ to a transitive partition pair produces a non-transitive partition pair.

Example 3. *Consider the graph in Figure 2(a) and the partition pair $\langle \Sigma, P\rangle$ with $\Sigma = \{\alpha, \beta, \gamma\}$ as depicted and $P = \mathcal{I}(\Sigma)$. Let $\langle \Pi, Q\rangle = \sigma(\langle \Sigma, P\rangle)$, then:*

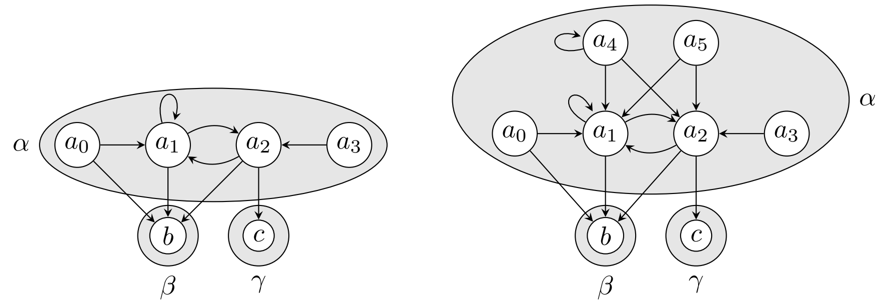
$$\Pi = \{\alpha_1, \alpha_2, \alpha_3, \beta, \gamma\} \qquad Q = \mathcal{I}(\Pi) \cup \{(\alpha_3, \alpha_1), (\alpha_1, \alpha_2)\}$$

where $\alpha_1 = \{a_0, a_1\}$, $\alpha_2 = \{a_2\}$ and $\alpha_3 = \{a_3\}$. □

Our final counterexample shows that σ is not suitable for computing the solution of the GCPP, and is constructed by embedding Counterexample 2 in Example 3, such that the first application of σ produces a non-transitive partition pair on which σ is not well-defined.

Counterexample 4. *Consider the graph in Figure 2(b) and the partition pair $\langle \Sigma, P\rangle$ with $\Sigma = \{\alpha, \beta, \gamma\}$ as depicted and $P = \mathcal{I}(\Sigma)$. Let $\langle \Pi, Q\rangle = \sigma(\langle \Sigma, P\rangle)$, then:*

$$\Pi = \{\alpha_1, \alpha_2, \alpha_3, \beta, \gamma\} \qquad Q = \mathcal{I}(\Pi) \cup \{(\alpha_3, \alpha_1), (\alpha_1, \alpha_2)\}$$



(a) (b)

Figure 2: (a) Example for which σ produces a non-transitive relation Q and (b) counterexample for correctness of σ.

where $\alpha_1 = \{a_0, a_1\}$*,* $\alpha_2 = \{a_2\}$ *and* $\alpha_3 = \{a_3, a_4, a_5\}$*. Now, in* $\langle \Pi, Q\rangle$ *the block* α_3 *has to be split, because* $\alpha_3 \rightarrow_\exists \alpha_3$ *but* $\neg\exists\delta \in \Pi \,.\, ((\alpha_3, \delta) \in Q \wedge \alpha_3 \rightarrow_\forall \delta))$*. There are two candidate partition pairs for* $\sigma(\langle \Pi, Q\rangle)$*:* α_3 *can be split into either* $\alpha_{3,0} = \{a_4\}$ *and* $\alpha_{3,1} = \{a_3, a_5\}$ *or* $\alpha'_{3,0} = \{a_4, a_5\}$ *and* $\alpha'_{3,1} = \{a_3\}$*. However, neither of these is greater than the other, so a unique* $\leq$*-maximal partition pair does not exist.* $\square$

When splitting α_3 in Counterexample 4, the REFINE$_{\text{GPP}}$ function of algorithm PA$_{\text{GPP}}$ splits the block into $\alpha_{3,0}$ and $\alpha_{3,1}$. Observe that this is wrong: a_4 and a_5 should not end up in different equivalence classes because $a_4 \rightleftarrows a_5$. This split also results in UPDATE$_{\text{GPP}}$'s returning a cyclic relation. In the subsequent iteration of PA$_{\text{GPP}}$, the execution of REFINE$_{\text{GPP}}$ then fails because there is no reverse topological sorting of the partition w.r.t. the cyclic relation (line 4).

5 An Auxiliary Fixed-Point Operator

In this section we introduce a fixed-point operator ρ to solve the GCPP and prove its correctness. The definition of ρ is straightforward: it is based directly on the stability condition of Section 2.

We emphasise that ρ is not intended to be an improvement over the σ operator of Section 3 in any way: it is a less advanced operator than σ aimed to be. The purpose of σ was to compute the solution to the GCPP efficiently, while ρ gives rise to an algorithm that has an inferior time complexity of $\mathcal{O}(S^3T)$ where S is the number of equivalence classes of the GCPP solution and T the number of transitions of the input graph.

Namely, the complexity analysis of [8] uses that, as long as no fixed point is reached, in each refinement-update step the refinement of the partition will be non-trivial, *i.e.* the number of blocks increases. As a consequence, there will be at most S refinement-update steps before the algorithm terminates. Such an analysis is not appropriate for ρ: applying ρ repeatedly could involve many steps in which the partition does not change. Consequently, the number of iterations of the algorithm is bounded merely by the size of a relation on the eventual partition, *i.e.* by S^2.

The sole purpose of ρ is to serve as an auxiliary operator for establishing the correctness of the algorithm that we present in Section 6. That algorithm has the same time complexity as $\mathrm{PA}_{\mathrm{GPP}}$ and does not correspond to any fixed-point operator, as we show in the same section.

Definition 5 (Operator ρ)**.** Let $\langle\Sigma, P\rangle$ be a transitive partition pair over a graph $(N, \rightarrow)$. Then $\rho(\langle\Sigma, P\rangle)$ is the $\leq$-largest partition pair $\langle\Pi, Q\rangle \leq \langle\Sigma, P\rangle$ that satisfies

$$(1)\ \forall\alpha, \beta \in \Pi\ .\ \forall\gamma \in \Sigma\ .\ ((\alpha, \beta) \in Q \wedge \alpha \rightarrow_{\exists} \gamma\ \Rightarrow\ \exists\delta \in \Sigma\ .\ ((\gamma, \delta) \in P \wedge \beta \rightarrow_{\forall} \delta)).$$

Alternatively, ρ could be defined just like σ of Definition 4.11, but insisting that its input partition pair is transitive, and omitting clause (c). It is not hard to check that this definition is equivalent to the one above. The correctness of Definition 5 is ensured by the following.

Proposition 6. *Let $\langle\Sigma, P\rangle$ be a transitive partition pair over a graph $(N, \rightarrow)$. Then there exists a $\leq$-largest partition pair $\langle\Pi, Q\rangle \leq \langle\Sigma, P\rangle$ that satisfies* (1)*. Moreover, Q is transitive.*

Proof. Define the relation $\sqsubseteq\ \subseteq N \times N$ by $a \sqsubseteq b$ iff

$$\exists(\alpha, \beta) \in P\ .\ a \in \alpha \wedge b \in \beta \wedge \forall\gamma \in \Sigma\ .\ (a \rightarrow \gamma \Rightarrow \exists\delta \in \Sigma\ .\ ((\gamma, \delta) \in P \wedge b \rightarrow \delta)).$$

Using the reflexivity and transitivity of P, this relation is a preorder. Take $\langle\Pi, Q\rangle := \mathsf{PP}(\sqsubseteq)$, as defined in Section 2. So Q is transitive. By construction, $\langle\Pi, Q\rangle \leq \langle\Sigma, P\rangle$. It is not hard to check that Π satisfies (1).[3]

Now let $\langle\Pi', Q'\rangle$ be another partition pair with $\langle\Pi', Q'\rangle \leq \langle\Sigma, P\rangle$ that satisfies (1). Suppose $(\alpha, \beta) \in Q'$, $a \in \alpha$ and $b \in \beta$. Using (1) we find $a \sqsubseteq b$. Applying this insight to the case $\alpha = \beta$ we find that Π' is finer than Π. Applying it in general yields $Q' \subseteq Q(\Pi')$. Hence $\langle\Pi', Q'\rangle \leq \langle\Pi, Q\rangle$. □

Proposition 7. *The operator ρ is monotone with respect to $\leq$: if $\langle\Sigma, P\rangle$ and $\langle\Sigma', P'\rangle$ are transitive partition pairs with $\langle\Sigma, P\rangle \leq \langle\Sigma', P'\rangle$, then $\rho(\langle\Sigma, P\rangle) \leq \rho(\langle\Sigma', P'\rangle)$.*

Proof. As $\rho(\langle\Sigma, P\rangle)$ satisfies (1) w.r.t. $\langle\Sigma, P\rangle$, it certainly satisfies (1) w.r.t. $\langle\Sigma', P'\rangle$. As $\rho(\langle\Sigma, P\rangle) \leq \langle\Sigma, P\rangle \leq \langle\Sigma', P'\rangle$ and $\rho(\langle\Sigma', P'\rangle)$ is the $\leq$-largest partition pair with $\rho(\langle\Sigma', P'\rangle) \leq \langle\Sigma', P'\rangle$ that satisfies (1), it follows that $\rho(\langle\Sigma, P\rangle) \leq \rho(\langle\Sigma', P'\rangle)$. □

Since $\rho(\langle\Sigma, P\rangle) \leq \langle\Sigma, P\rangle$ and $\leq$ is a partial order on a finite set, we obtain:

Proposition 8. *Let $\langle\Sigma, P\rangle$ be a transitive partition pair over a graph. Then for some $n \geq 0$, $\rho^{n+1}(\langle\Sigma, P\rangle) = \rho^{n}(\langle\Sigma, P\rangle)$, i.e. repeated application of ρ leads to a fixed point.* □

The solution to the GCPP over an input graph G and an initial partition pair $\langle\Sigma, P\rangle$ over G can be obtained by repeatedly applying ρ to $\langle\Sigma, P^{+}\rangle$. The following lemmata say that as soon as a fixed point is reached, the resulting partition pair is stable. Moreover, each of the intermediate partition pairs is larger than or equal to the solution of the GCPP. It then follows that the obtained fixed point is in fact the solution to the GCPP.

[3] Suppose $(\alpha, \beta) \in Q$ and $\alpha \rightarrow_{\exists} \gamma$ for $\gamma \in \Sigma$. Then $\exists a \in \alpha\ .\ a \rightarrow \gamma$. Take that a, and a $b' \in \beta$. As $a \sqsubseteq b'$, we have $\exists\delta' \in \Sigma\ .\ ((\gamma, \delta') \in P \wedge b' \rightarrow \delta'))$. Hence $\beta \rightarrow_{\exists} \delta'$. As P is a partial order on a finite set, let δ be a P-maximal element of Σ larger than δ' such that $\beta \rightarrow_{\exists} \delta$, *i.e.* $(\delta', \delta) \in P$ and $\forall\varepsilon \in \Sigma\ .\ (\delta, \varepsilon) \in P \wedge \beta \rightarrow_{\exists} \varepsilon \Rightarrow \varepsilon = \delta$. Note that $(\gamma, \delta) \in P$. As $\beta \rightarrow_{\exists} \delta$, $\exists b_0 \in \beta\ .\ b_0 \rightarrow \delta$. For any $b \in \beta$ we have $b_0 \sqsubseteq b$, so $\exists\varepsilon_b \in \Sigma\ .\ ((\delta, \varepsilon_b) \in P \wedge b \rightarrow \varepsilon_b))$. It must be that $\varepsilon_b = \delta$. Hence $\beta \rightarrow_{\forall} \delta$.

Lemma 9. *Let $\langle \Sigma, P \rangle$ be a transitive partition pair over a graph $(N, \rightarrow)$. Then $\rho(\langle \Sigma, P \rangle) = \langle \Sigma, P \rangle$ if and only if $\langle \Sigma, P \rangle$ is stable with respect to $\rightarrow$.*

Proof. Because $\rho(\langle \Sigma, P \rangle)$ is the $\leq$-largest partition pair satisfying (1), we have that $\rho(\langle \Sigma, P \rangle) = \langle \Sigma, P \rangle$ if and only if $\langle \Sigma, P \rangle$ satisfies (1) w.r.t. itself, which is equivalent to stability w.r.t. $\rightarrow$. $\square$

Lemma 10. *Let $\langle \Sigma, P \rangle$ and $\langle \Pi, Q \rangle$ be partition pairs over a graph G, with Q transitive, and let $\langle \Xi, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Sigma, P \rangle$. If $\langle \Xi, \preceq \rangle \leq \langle \Pi, Q \rangle$ then $\langle \Xi, \preceq \rangle \leq \rho(\langle \Pi, Q \rangle)$.*

Proof. By Lemma 9 $\rho(\langle \Xi, \preceq \rangle) = \langle \Xi, \preceq \rangle$. Assuming that $\langle \Xi, \preceq \rangle \leq \langle \Pi, Q \rangle$, the statement now follows from Proposition 7. $\square$

Theorem 11. *Let $\langle \Sigma, P \rangle$ be a partition pair over a graph $G = (N, \rightarrow)$ and $\langle \Xi, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Sigma, P \rangle$. Let $n \geq 0$ be such that $\rho^{n+1}(\langle \Sigma, P^+ \rangle) = \rho^n(\langle \Sigma, P^+ \rangle)$. Then $\rho^n(\langle \Sigma, P^+ \rangle) = \langle \Xi, \preceq \rangle$.*

Proof. Note that n exists by Proposition 8. We prove that $\langle \Xi, \preceq \rangle \leq \rho^n(\langle \Sigma, P^+ \rangle)$ and $\rho^n(\langle \Sigma, P^+ \rangle) \leq \langle \Xi, \preceq \rangle$.

- $\langle \Xi, \preceq \rangle \leq \rho^n(\langle \Sigma, P^+ \rangle)$: By definition $\langle \Xi, \preceq \rangle \leq \langle \Sigma, P^+ \rangle$. Applying Lemma 10 n times gives us $\langle \Xi, \preceq \rangle \leq \rho^n(\langle \Sigma, P^+ \rangle)$.
- $\rho^n(\langle \Sigma, P^+ \rangle) \leq \langle \Xi, \preceq \rangle$: Obviously $\rho^n(\langle \Sigma, P^+ \rangle) \leq \langle \Sigma, P^+ \rangle$ and by Lemma 9 $\rho^n(\langle \Sigma, P^+ \rangle)$ is stable w.r.t. $\rightarrow$. By definition $\langle \Xi, \preceq \rangle$ is the $\leq$-largest partition pair that has these properties. Hence $\rho^n(\langle \Sigma, P^+ \rangle) \leq \langle \Xi, \preceq \rangle$. $\square$

6 A Correct and Efficient Algorithm

Algorithm 3 The repaired partitioning algorithm: PA($(N, \rightarrow), \langle \Sigma, P \rangle$)

```
1: Σ_1 := REFINE(Σ, P);
2: P_1 := UPDATE_GPP(Σ, P, Σ_1);
3: change := ⊤; i := 1;
4: while change do
5:     change := ⊥;
6:     Σ_{i+1} := REFINE(Σ_i, P_i);
7:     P_{i+1} := UPDATE_GPP(Σ_i, P_i, Σ_{i+1});
8:     i := i + 1;
9: end while
```

Our repaired partitioning algorithm is called PA, see Algorithm 3. The variable *change* and the input graph $(N, \rightarrow)$ have global scope: they can be accessed from any function. Note however, that UPDATE$_{\text{GPP}}$ does not access *change*.

Our corrections of the algorithm are two. Firstly, it is ensured that at least two refinement-update steps are taken before the algorithm terminates (lines 1 and 2). The necessity of this (minor) correction is explained in Section 6.1. Secondly, the most important error — the one resulting from the incorrect σ operator — is repaired by the new REFINE function, Algorithm 4. It contains a few minor improvements over REFINE$_{\text{GPP}}$: using list notations for variable *Sort* and preventing empty blocks from

Algorithm 4 The repaired refine function: $\textsc{Refine}(\Sigma, P)$

1: $\Pi := \Sigma$;
2: **for all** $\alpha \in \Pi$ **do** $\mathit{Stable}(\alpha) := \emptyset$; **end for**
3: **for all** $\gamma \in \Sigma$ **do** $\mathit{Row}(\gamma) := \{\gamma' \mid (\gamma, \gamma') \in P\}$; **end for**
4: Let *Sort* be a reverse topological sorting of Σ w.r.t. P;
5: **while** $\mathit{Sort} \neq []$ **do**
6: $\quad\gamma := \mathit{head}(\mathit{Sort})$;
7: $\quad A := \emptyset$;
8: $\quad$**for all** $\alpha \in \Pi$, $\alpha \rightarrow_\exists \gamma$ **do**
9: $\quad\quad$**if** $\mathit{Stable}(\alpha) \cap \mathit{Row}(\gamma) = \emptyset$ **then**
10: $\quad\quad\quad\alpha_1 := \alpha \cap \rightarrow^{-1}(\gamma)$;
11: $\quad\quad\quad\alpha_2 := \alpha \setminus \alpha_1$;
12: $\quad\quad\quad\Pi := \Pi \setminus \{\alpha\}$;
13: $\quad\quad\quad A := A \cup \{\alpha_1\}$;
14: $\quad\quad\quad\mathit{Stable}(\alpha_1) := \mathit{Stable}(\alpha) \cup \{\gamma\}$;
15: $\quad\quad\quad$**if** $\alpha_2 \neq \emptyset$ **then**
16: $\quad\quad\quad\quad\mathit{change} := \top$;
17: $\quad\quad\quad\quad A := A \cup \{\alpha_2\}$;
18: $\quad\quad\quad\quad\mathit{Stable}(\alpha_2) := \mathit{Stable}(\alpha)$;
19: $\quad\quad\quad$**end if**
20: $\quad\quad$**else**
21: $\quad\quad\quad\mathit{Stable}(\alpha) := \mathit{Stable}(\alpha) \cup \{\gamma\}$;
22: $\quad\quad$**end if**
23: $\quad$**end for**
24: $\quad\Pi := \Pi \cup A$;
25: $\quad\mathit{Sort} := \mathit{tail}(\mathit{Sort})$;
26: **end while**
27: **return** Π;

being added to Π. However, the actual correction is in line 21: if for some $\gamma \in \Sigma$ and $\alpha \in \Pi$ with $\alpha \rightarrow_\exists \gamma$ we have $\mathit{Stable}(\alpha) \cap \mathit{Row}(\gamma) \neq \emptyset$ then we add γ to $\mathit{Stable}(\alpha)$.

We use the ρ operator of Section 5 to prove correctness of PA in Section 6.2. Its space and time complexities are the same as for $\mathrm{PA}_{\mathrm{GPP}}$: no additional space is needed and the corrections do not increase the time complexity. Finally, in Section 6.3 we show that there is no fixed-point operator that captures the refinement performed by our REFINE function.

6.1 The Correction of a Minor Mistake

Apart from the error in $\mathrm{PA}_{\mathrm{GPP}}$ that results from the incorrect σ operator, we found another, minor mistake in the algorithm. We describe it in this section and propose a solution. The mistake is shown by the following example.

Example 12. *Consider the graph $G = (N, \rightarrow)$ in Figure 3 and the partition pair $\langle \Sigma, P \rangle$ with $\Sigma = \{\alpha, \beta\}$ as depicted and $P = \mathcal{I}(\Sigma) \cup \{(\alpha, \beta)\}$. Observe that the solution to the GCPP over G and $\langle \Sigma, P \rangle$ is $\langle \Xi, \preceq \rangle$ with $\Xi = \{\alpha_0, \alpha_1, \beta\}$ and $\preceq = \mathcal{I}(\Xi) \cup \{(\alpha_1, \alpha_0)\}$ where $\alpha_i = \{a_i\}$. After the first iteration of $\mathrm{PA}_{\mathrm{GPP}}(G, \langle \Sigma, P \rangle)$, we have $\Sigma_1 = \Sigma_0 = \Sigma$ and $P_1 = \mathcal{I}(\Sigma)$. The algorithm then terminates because $\mathit{change} = \bot$, and $\langle \Sigma_1, P_1 \rangle$ is its answer to the GCPP over G and $\langle \Sigma, P \rangle$. Obviously*

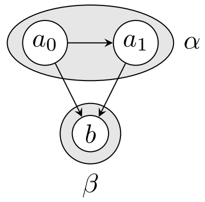


Figure 3: Example for which the algorithm $\mathrm{PA}_{\mathrm{GPP}}$ terminates prematurely.

$\langle \Sigma_1, P_1 \rangle \neq \langle \Xi, \preceq \rangle$, *so this answer is wrong.* □

The correctness of $\mathrm{PA}_{\mathrm{GPP}}$ hinges on the theory that whenever $\textsc{Refine}_{\mathrm{GPP}}(\Pi, Q, \mathit{change})$ returns its input partition Π, and thus fails to split any block in Π, then also the relation Q will be unaffected by $\textsc{Update}_{\mathrm{GPP}}$, *i.e.* $\textsc{Update}_{\mathrm{GPP}}(\Pi, Q, \Pi)$ returns Q. This theory is the upshot of Theorem 4.15 in [8] and is essential in the complexity analysis of the algorithm. However, the above example shows that it does not hold in general.

In the next section we show that this theory does hold under the condition that Q itself is obtained as output of $\textsc{Update}_{\mathrm{GPP}}$ (Proposition 14). Therefore, this error in $\mathrm{PA}_{\mathrm{GPP}}$ can be fixed, without violating the complexity analysis, by insisting that at least two refinement-update steps are performed prior to termination.

6.2 Correctness of PA

From here on we will use the correctness of the function $\textsc{Update}_{\mathrm{GPP}}$, as established by Gentilini *et al.* [9]. This correctness can be summarised as follows:

Proposition 13. *Let $\langle \Sigma, P \rangle$ be a partition pair over a graph $(N, \rightarrow)$, and Π be a partition over N that is finer than Σ. Then there exists a unique relation $Q \subseteq P(\Pi)$ satisfying condition (2σ) of Definition 4.11. Moreover, this relation is returned by* $\textsc{Update}_{\mathrm{GPP}}(\Sigma, P, \Pi)$.

Proof. The union of all relations $Q \subseteq P(\Pi)$ such that (b) and (c) hold for all $(\alpha, \beta) \in Q$ is itself a relation with these properties. The last claim has been established in [9]. □

Using this, we obtain the result promised in Section 6.1: the following proposition implies that if a call to $\textsc{Refine}$ in the while-loop of PA does not split any blocks, then the subsequent call to $\textsc{Update}_{\mathrm{GPP}}$ will return its input relation. The requirement that this relation has been computed by a previous call to $\textsc{Update}_{\mathrm{GPP}}$ is guaranteed by line 2.

Proposition 14. *Let $\langle \Sigma, P \rangle$ and $\langle \Pi, Q \rangle$ be partition pairs over a graph such that Π is finer than Σ and $\textsc{Update}_{\mathrm{GPP}}(\Sigma, P, \Pi)$ returns Q. Then $\textsc{Update}_{\mathrm{GPP}}(\Pi, Q, \Pi)$ also returns Q.*

Proof. By Proposition 13, $\textsc{Update}_{\mathrm{GPP}}(\Pi, Q, \Pi)$ returns the largest relation $Q' \subseteq Q(\Pi)$ satisfying conditions (b) and (c) of Definition 4.11 w.r.t. Π, Q and Π (*i.e.* substituting Q', Π, Q and Π for Q, Σ, P and Π in these conditions, respectively). We have to prove that $Q' = Q$. As $Q = Q(\Pi)$ it suffices to show that Q satisfies (b) and (c) with Π substituted for Σ and Q for P. Under these substitutions (b) becomes equal to (c). By Proposition 13 applied to $\textsc{Update}_{\mathrm{GPP}}(\Sigma, P, \Pi)$, Q satisfies this condition. □

Let $\langle \Sigma_i, P_i \rangle_{1 \leq i \leq k}$ be the sequence of partition pairs produced by PA. The following proposition says that every P_i is acyclic and that the sequence is decreasing. The former implies that PA will never deadlock due to the inability to find a reverse topological sorting (see line 4 of REFINE). The latter implies that the algorithm terminates.

Proposition 15. *Let $\langle \Sigma, P \rangle$ be a partition pair over a graph $(N, \rightarrow)$,* REFINE(Σ, P) *return* Π *and* UPDATE$_{\text{GPP}}(\Sigma, P, \Pi)$ *return* Q*. Then* $\langle \Pi, Q \rangle$ *is a partition pair with* $\langle \Pi, Q \rangle \leq \langle \Sigma, P \rangle$.

Proof. From Algorithm 4 and the fact that Σ is a partition, it is not hard to see that Π is a partition that is, moreover, finer than Σ. Also, by Proposition 13 we have that $Q \subseteq P(\Pi)$. Hence $\langle \Pi, Q \rangle \leq \langle \Sigma, P \rangle$. To prove that the pair $\langle \Pi, Q \rangle$ is a partition pair, we need to prove reflexivity and acyclicity of Q. Using reflexivity of P and $P(\Pi)$, the identity relation $\mathcal{I}(\Pi)$ trivially satisfies conditions (b) and (c) of Definition 4.11. Hence Proposition 13 implies that $\mathcal{I}(\Pi) \subseteq Q$, *i.e.* Q is reflexive.

Suppose Q contains a cycle: there are mutually distinct $\alpha_0, \ldots, \alpha_{n-1} \in \Pi$ for $n > 1$ such that $(\alpha_i, \alpha_{i+1 \bmod n}) \in Q$ for $0 \leq i < n$. By acyclicity of P, it must be that these α_i are all subsets of the same block $\alpha \in \Sigma$. Let $\gamma \in \Sigma$ and $\alpha' \subseteq \alpha$ be the first blocks considered in an iteration of REFINE's main for-loop (line 8) such that γ splits α' into an α'_1 and an α'_2 such that $\alpha_i \subseteq \alpha'_1$ and $\alpha_j \subseteq \alpha'_2$ for some $0 \leq i, j < n$. Then $\mathit{Stable}(\alpha') \cap \mathit{Row}(\gamma) = \emptyset$. For any $0 \leq k < n$ we have either $\alpha_k \rightarrow_\forall \gamma$ or $\alpha_k \not\rightarrow_\exists \gamma$, and both possibilities occur. Take $0 \leq i < n$ such that $\alpha_{i-1 \bmod n} \rightarrow_\forall \gamma$ and $\alpha_i \not\rightarrow_\exists \gamma$. By Proposition 13, Q satisfies (b) of Definition 4.11. Hence $\exists \gamma' \in \Sigma \,.\, ((\gamma, \gamma') \in P \wedge \alpha_i \rightarrow_\exists \gamma'))$. As $(\gamma, \gamma') \in P$, in REFINE's while-loop γ' is considered prior to γ. Consider the unique iteration of REFINE's main for-loop (line 8) involving γ' and an α'' with $\alpha' \subseteq \alpha'' \subseteq \alpha$ — observe that $\alpha'' \rightarrow_\exists \gamma'$. At the end of that iteration we have obtained a block α''' with $\alpha' \subseteq \alpha''' \subseteq \alpha''$ and $\gamma' \in \mathit{Stable}(\alpha''')$. It follows that at the later iteration involving γ and α' we have $\gamma' \in \mathit{Stable}(\alpha') \cap \mathit{Row}(\gamma)$, which is a contradiction. ☐

Corollary 16. *For any graph G and any partition pair $\langle \Sigma, P \rangle$ over G, the algorithm* PA$(G, \langle \Sigma, P \rangle)$ *terminates.* ☐

Lemma 17. *The following predicate is an invariant for the while-loop of Algorithm 4:*

$$\forall \beta \in \Pi \cup A \,.\, \forall \varepsilon \in \mathit{Stable}(\beta) \,.\, \exists \delta \in \Sigma \,.\, ((\varepsilon, \delta) \in P^+ \wedge \beta \rightarrow_\forall \delta)\,.$$

Proof. The predicate holds trivially after the initialisation of the *Stable*-sets in line 2. The only points where it could be violated are at lines 14, 18 and 21. For $\varepsilon \neq \gamma$ lines 14 and 18 are harmless because if $\alpha_i \subseteq \alpha$ and $\alpha \rightarrow_\forall \delta$ then certainly $\alpha_i \rightarrow_\forall \delta$. For $\varepsilon = \gamma$ and $\beta = \alpha_1$, at line 14 the predicate holds by construction of α_1, taking $\delta := \gamma$. Finally, line 21 is only executed when there is an $\varepsilon \in \mathit{Stable}(\alpha) \cap \mathit{Row}(\gamma)$. As $(\gamma, \varepsilon) \in P$, the predicate holds for γ and α because it held already for ε and α. ☐

Lemma 18. *Let $\langle \Sigma, P \rangle$ be a partition pair over a graph $(N, \rightarrow)$ and* REFINE(Σ, P) *return* Π*. Then:*

$$\forall \alpha \in \Pi \,.\, \forall \gamma \in \Sigma \,.\, (\alpha \rightarrow_\exists \gamma \;\Rightarrow\; \exists \delta \in \Sigma \,.\, ((\gamma, \delta) \in P^+ \wedge \alpha \rightarrow_\forall \delta))\,.$$

Proof. Let $\alpha \in \Pi$ and $\gamma \in \Sigma$ such that $\alpha \rightarrow_\exists \gamma$. In the computation of Π, take the unique iteration of REFINE's main for-loop (line 8) in which γ and an α' are considered with $\alpha \subseteq \alpha'$. Then $\alpha' \rightarrow_\exists \gamma$ and there are two cases:

- $\mathit{Stable}(\alpha') \cap \mathit{Row}(\gamma) = \emptyset$: Then α' is split into α_1 and α_2 such that $\alpha_1 \rightarrow_\forall \gamma$ and $\alpha_2 \not\rightarrow_\exists \gamma$. It must be that $\alpha \subseteq \alpha_1$. Then $\alpha \rightarrow_\forall \gamma$ and $(\gamma, \gamma) \in P^+$.
- $\mathit{Stable}(\alpha') \cap \mathit{Row}(\gamma) \neq \emptyset$: Then γ is added to $\mathit{Stable}(\alpha')$. Lemma 17 gives us $\exists \delta \in \Sigma \;.\; ((\gamma, \delta) \in P^+ \wedge \alpha' \rightarrow_\forall \delta)$. As $\alpha \subseteq \alpha'$ we have $\alpha \rightarrow_\forall \delta$. □

Lemma 19. *Let $\langle \Sigma, P\rangle$ and $\langle \Pi, Q\rangle$ be partition pairs over a graph $(N, \rightarrow)$ such that $\langle \Pi, Q\rangle \leq \langle \Sigma, P\rangle$ and let P be transitive. If $\langle \Pi, Q\rangle$ satisfies* (1) *of Definition 5 w.r.t. $\langle \Sigma, P\rangle$ then so does $\langle \Pi, Q^+\rangle$.*

Proof. Suppose $\langle \Pi, Q\rangle$ satisfies (1) w.r.t. $\langle \Sigma, P\rangle$ and take $(\alpha, \beta) \in Q^+$ and $\gamma \in \Sigma$ such that $\alpha \rightarrow_\exists \gamma$. There are $\alpha_0, \ldots, \alpha_n \in \Pi$ for $n \geq 0$ such that $\alpha = \alpha_0, \beta = \alpha_n$ and $(\alpha_i, \alpha_{i+1}) \in Q$ for $0 \leq i < n$. Applying (1) n times we obtain $\delta_1, \ldots, \delta_n \in \Sigma$ such that $\alpha_i \rightarrow_\forall \delta_i$ (and thus $\alpha_i \rightarrow_\exists \delta_i$) for $1 \leq i \leq n$, $(\gamma, \delta_1) \in P$ and $(\delta_i, \delta_{i+1}) \in P$ for $1 \leq i < n$. Hence $\beta \rightarrow_\forall \delta_n$ and $(\gamma, \delta_n) \in P$ by transitivity of P. □

The following lemmata state that REFINE and UPDATE$_{\text{GPP}}$ converge towards a fixed point at least as fast as ρ without ever diverging from the path towards the GCPP solution. In combination with the monotony of ρ (Proposition 7) this implies the correctness of our algorithm.

Lemma 20. *Let $\langle \Sigma, P\rangle$ be a partition pair over a graph $(N, \rightarrow)$,* REFINE(Σ, P) *return Π, and* UPDATE$_{\text{GPP}}(\Sigma, P, \Pi)$ *return Q. Then $\langle \Pi, Q^+\rangle \leq \rho(\langle \Sigma, P^+\rangle)$.*

Proof. By Proposition 15, $\langle \Pi, Q\rangle$ is a partition pair with $\langle \Pi, Q\rangle \leq \langle \Sigma, P\rangle \leq \langle \Sigma, P^+\rangle$. By Definition 5, $\rho(\langle \Sigma, P^+\rangle)$ is the $\leq$-largest partition pair smaller than $\langle \Sigma, P^+\rangle$ that satisfies (1) w.r.t. $\langle \Sigma, P^+\rangle$. So the statement follows if we prove that $\langle \Pi, Q^+\rangle$ satisfies (1) w.r.t. $\langle \Sigma, P^+\rangle$. By Lemma 19 it suffices to show that $\langle \Pi, Q\rangle$ satisfies (1) w.r.t. $\langle \Sigma, P^+\rangle$. Let $(\alpha, \beta) \in Q$ and $\alpha \rightarrow_\exists \gamma$ for $\gamma \in \Sigma$. Using Lemma 18, take $\delta \in \Sigma$ such that $(\gamma, \delta) \in P^+$ and $\alpha \rightarrow_\forall \delta$. By Proposition 13, $\exists \delta' \in \Sigma \;.\; (\delta, \delta') \in P \wedge \beta \rightarrow_\exists \delta'$. For that δ', by Lemma 18, $\exists \gamma' \in \Sigma \;.\; (\delta', \gamma') \in P^+ \wedge \beta \rightarrow_\forall \gamma'$. For this γ' it holds that $(\gamma, \gamma') \in P^+$. Hence $\langle \Pi, Q\rangle$ satisfies (1) w.r.t. $\langle \Sigma, P^+\rangle$. □

Lemma 21. *Let $\langle \Sigma, P\rangle$ and $\langle \Pi, Q\rangle$ be partition pairs over a graph $G = (N, \rightarrow)$, $\langle \Xi, \preceq\rangle$ be the solution of the GCPP over G and $\langle \Sigma, P\rangle$, and $\langle \Xi, \preceq\rangle \leq \langle \Pi, Q\rangle$. Let* REFINE$(\Pi,Q)$ *return Π' and* UPDATE$_{\text{GPP}}(\Pi,Q,\Pi')$ *return Q'. Then $\langle \Xi,\preceq\rangle \leq \langle \Pi',Q'\rangle$.*

Proof. We have to prove that (A) Ξ is finer than Π' and (B) $\preceq \subseteq Q'(\Xi)$.

Ad (A). Let $\alpha \in \Xi$ and $\alpha_\Pi \in \Pi$ such that $\alpha \subseteq \alpha_\Pi$. By contradiction, suppose there is no $\alpha' \in \Pi'$ such that $\alpha \subseteq \alpha'$. Hence, there are $a_1, a_2 \in \alpha$ such that REFINE at some point separates $a_1 \in \alpha_\Pi$ from $a_2 \in \alpha_\Pi$. Let $\alpha'_\Pi \subseteq \alpha_\Pi$ be such that $a_1, a_2 \in \alpha'_\Pi$ and a_1 and a_2 got separated when α'_Π was split by a block $\gamma_\Pi \in \Pi$. Hence $\mathit{Stable}(\alpha'_\Pi) \cap \mathit{Row}(\gamma_\Pi) = \emptyset$.

Consider the case where $a_1 \rightarrow \gamma_\Pi$ and $a_2 \not\rightarrow \gamma_\Pi$. The case with $a_1 \not\rightarrow \gamma_\Pi$ and $a_2 \rightarrow \gamma_\Pi$ is fully symmetrical. Let $\gamma \in \Xi$ be such that $\gamma \subseteq \gamma_\Pi$ and $a_1 \rightarrow \gamma$. As $\alpha \rightarrow_\exists \gamma$ and $\langle \Xi, \preceq\rangle$ is stable w.r.t. $\rightarrow$, there must be a $\delta \in \Xi$ with $\gamma \preceq \delta$ and $\alpha \rightarrow_\forall \delta$. Let $\delta_\Pi \in \Pi$ be such that $\delta \subseteq \delta_\Pi$. Then $(\gamma_\Pi, \delta_\Pi) \in Q$, using that $\langle \Xi, \preceq\rangle \leq \langle \Pi, Q\rangle$. So $\delta_\Pi \in \mathit{Row}(\gamma_\Pi)$ and δ_Π is before γ_Π in the reverse topological sorting of Π w.r.t. Q. As $a_2 \not\rightarrow \gamma_\Pi$ we have $\alpha \not\rightarrow_\forall \gamma_\Pi$, yet $\alpha \rightarrow_\forall \delta_\Pi$, hence $\gamma_\Pi \neq \delta_\Pi$. Let $\alpha''_\Pi \subseteq \alpha_\Pi$ be the block containing a_1 and a_2 when blocks were split w.r.t. δ_Π by REFINE. Observe that $\alpha''_\Pi \rightarrow_\exists \delta_\Pi$, so there were two cases:

- $\mathit{Stable}(\alpha''_\Pi) \cap \mathit{Row}(\delta_\Pi) = \emptyset$: Then α''_Π may have been split, but this did not separate a_1 and a_2. Then $\alpha'_\Pi \subseteq (\alpha''_\Pi \cap \rightarrow^{-1}(\delta_\Pi))$ and hence $\delta_\Pi \in \mathit{Stable}(\alpha'_\Pi)$.
- $\mathit{Stable}(\alpha''_\Pi) \cap \mathit{Row}(\delta_\Pi) \neq \emptyset$: Then δ_Π was added to $\mathit{Stable}(\alpha''_\Pi)$ (line 21) and because $\alpha'_\Pi \subseteq \alpha''_\Pi$ we have $\delta_\Pi \in \mathit{Stable}(\alpha'_\Pi)$.

In both cases we have that $\delta_\Pi \in \mathit{Stable}(\alpha'_\Pi) \cap \mathit{Row}(\gamma_\Pi)$, which contradicts the fact that $\mathit{Stable}(\alpha'_\Pi) \cap \mathit{Row}(\gamma_\Pi) = \emptyset$.

Ad (B). Let $Q'_\preceq := \{(\alpha,\beta) \in \Pi' \times \Pi' \mid \exists(\alpha_\Xi, \beta_\Xi) \in \preceq .\ \alpha_\Xi \subseteq \alpha \wedge \beta_\Xi \subseteq \beta\}$. We will show that $Q'_\preceq \subseteq Q'$, which immediately yields $\preceq \subseteq Q'_\preceq(\Xi) \subseteq Q'(\Xi)$.

To this end, using Proposition 13, we establish that $Q'_\preceq \subseteq Q(\Pi')$ and any pair $(\alpha,\beta) \in Q'_\preceq$ satisfies conditions (b) and (c) of Definition 4.11, reading Π, Q, Π' and $Q'_\preceq$ for Σ, P, Π and Q, respectively.

- $Q'_\preceq \subseteq Q(\Pi')$: Let $(\alpha,\beta) \in Q'_\preceq$. Take $\alpha_\Pi, \beta_\Pi \in \Pi$ such that $\alpha \subseteq \alpha_\Pi$ and $\beta \subseteq \beta_\Pi$. Because $\preceq \subseteq Q(\Xi)$ we have $(\alpha_\Pi, \beta_\Pi) \in Q$, and hence $(\alpha,\beta) \in Q(\Pi')$.
- *Condition* (b): Let $(\alpha,\beta) \in Q'_\preceq$ and $\gamma \in \Pi$ such that $\alpha \rightarrow_\forall \gamma$. Take $\alpha_\Xi, \beta_\Xi \in \Xi$ such that $\alpha_\Xi \subseteq \alpha$, $\beta_\Xi \subseteq \beta$ and $\alpha_\Xi \preceq \beta_\Xi$. Also take $\gamma_\Xi \in \Xi$ such that $\gamma_\Xi \subseteq \gamma$ and $\alpha_\Xi \rightarrow_\exists \gamma_\Xi$. Because $\langle \Xi, \preceq \rangle$ is stable w.r.t. $\rightarrow$ we obtain a $\delta_\Xi \in \Xi$ such that $\gamma_\Xi \preceq \delta_\Xi$ and $\beta_\Xi \rightarrow_\forall \delta_\Xi$. Take $\delta \in \Pi$ such that $\delta_\Xi \subseteq \delta$. As $\preceq \subseteq Q(\Xi)$ we have $(\gamma,\delta) \in Q$. We also obtain $\beta \rightarrow_\exists \delta$.
- *Condition* (c): Let $(\alpha,\beta) \in Q'_\preceq$ and $\gamma \in \Pi'$ such that $\alpha \rightarrow_\forall \gamma$. Take $\alpha_\Xi, \beta_\Xi, \gamma_\Xi \in \Xi$ and obtain $\delta_\Xi \in \Xi$ exactly as above. Take $\delta \in \Pi'$ such that $\delta_\Xi \subseteq \delta$. We have $(\gamma,\delta) \in Q'_\preceq$ by construction. Again we obtain $\beta \rightarrow_\exists \delta$. □

Theorem 22. *Let $\langle \Sigma, P \rangle$ be a partition pair over a graph $G = (N, \rightarrow)$. Let k be the value of variable i upon termination of* PA$(G, \langle \Sigma, P^+ \rangle)$*. Then $\langle \Sigma_k, P_k \rangle$ is the solution of the GCPP over G and $\langle \Sigma, P \rangle$.*

Proof. Let the sequence of partition pairs $\langle \Sigma_i, P_i \rangle_{1 \leq i \leq k}$ be obtained by running PA$(G, \langle \Sigma, P^+ \rangle)$ until it terminates, implying that $k \geq 2$ and $\Sigma_k = \Sigma_{k-1}$. Proposition 14 yields $P_k = P_{k-1}$. Extend this sequence by defining $\langle \Sigma_0, P_0 \rangle := \langle \Sigma, P^+ \rangle$ and $\langle \Sigma_i, P_i \rangle := \langle \Sigma_k, P_k \rangle$ for $i > k$. Now for all $i \geq 0$ we have that REFINE(Σ_i, P_i) returns Σ_{i+1} and UPDATE$_{\text{GPP}}(\Sigma_i, P_i, \Sigma_{i+1})$ returns P_{i+1}. Let $\langle \Xi, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Sigma, P \rangle$. We need to show that $\langle \Sigma_k, P_k \rangle = \langle \Xi, \preceq \rangle$, for which we require the following properties:

(P1) $$\langle \Xi, \preceq \rangle \leq \langle \Sigma_i, P_i \rangle \text{ for all } i \geq 0$$

(P2) $$\langle \Sigma_i, P_i^+ \rangle \leq \rho^i(\langle \Sigma, P^+ \rangle) \text{ for all } i \geq 0.$$

Proof of (P1): By definition $\langle \Xi, \preceq \rangle \leq \langle \Sigma, P^+ \rangle = \langle \Sigma_0, P_0 \rangle$, and Lemma 21 yields $\langle \Xi, \preceq \rangle \leq \langle \Sigma_i, P_i \rangle$ for all $i > 0$, by induction on i.

Proof of (P2): By induction on i. If $i = 0$ then $\langle \Sigma_0, P_0^+ \rangle = \langle \Sigma, P^+ \rangle = \rho^0(\langle \Sigma, P^+ \rangle)$. For the inductive step, suppose $\langle \Sigma_i, P_i^+ \rangle \leq \rho^i(\langle \Sigma, P^+ \rangle)$. Then:

$$\langle \Sigma_{i+1}, P_{i+1}^+ \rangle \overset{\text{Lemma 20}}{\leq} \rho(\langle \Sigma_i, P_i^+ \rangle) \overset{\text{Proposition 7}}{\leq} \rho^{i+1}(\langle \Sigma, P^+ \rangle) \ .$$

Applying (P1) *and* (P2): By Proposition 8 and Theorem 11 there is an $n > 0$ such that $\rho^n(\langle \Sigma, P^+ \rangle) = \langle \Xi, \preceq \rangle$, so

$$\langle \Xi, \preceq \rangle \overset{\text{(P1)}}{\leq} \langle \Sigma_{n+1}, P_{n+1} \rangle \overset{\text{Prop. 15}}{\leq} \langle \Sigma_n, P_n \rangle \leq \langle \Sigma_n, P_n^+ \rangle \overset{\text{(P2)}}{\leq} \rho^n(\langle \Sigma, P^+ \rangle) = \langle \Xi, \preceq \rangle \ .$$

Thus $\langle \Sigma_{n+1}, P_{n+1} \rangle = \langle \Sigma_n, P_n \rangle$, so $k \leq n{+}1$, and $\langle \Sigma_k, P_k \rangle = \langle \Sigma_n, P_n \rangle = \langle \Xi, \preceq \rangle$. □

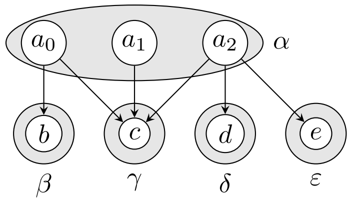


Figure 4: Example on which REFINE does not return a uniquely defined partition

6.3 No Fixed-Point Operator

We now show that there is no fixed-point operator that captures the partition refinement performed by REFINE, *i.e.* a function π such that for any partition pairs $\langle \Sigma, P \rangle$ and $\langle \Pi, Q \rangle$ with $\langle \Pi, Q \rangle = \pi(\langle \Sigma, P \rangle)$, REFINE$(\Sigma, P)$ returns Π. More specifically, we show that the partition returned by REFINE is not uniquely defined, but depends on the particular reverse topological sorting that is chosen in line 4.

Example 23. *Consider the graph $G = (N, \rightarrow)$ of Figure 4 and the partition pair $\langle \Sigma, P \rangle$ with $\Sigma = \{\alpha, \beta, \gamma, \delta, \varepsilon\}$ as depicted and $P = \mathcal{I}(\Sigma) \cup \{(\beta, \delta), (\delta, \gamma)\}$. Then $S = [\varepsilon, \gamma, \delta, \beta, \alpha]$ and $S' = [\gamma, \delta, \beta, \varepsilon, \alpha]$ are reverse topological sortings of Σ with respect to P. Let Π and Π' be the partitions returned by* REFINE(Σ, P) *on sortings S and S' respectively. Then $\Pi = \{\{a_0\}, \{a_1\}, \{a_2\}\}$ and $\Pi' = \{\{a_0, a_1\}, \{a_2\}\}$.* □

Similar to the construction of Counterexample 4, this example can be embedded in Example 3 to obtain an example with a transitive relation for which the partition after the second refinement depends on the chosen reverse topological sorting.

7 Conclusions

The correspondence between the simulation problem for finite, labelled graphs and the generalised coarsest partition problem (GCPP) for unlabelled graphs can be easily established. We have shown that the σ operator defined by Gentilini *et al.* [8] to solve the GCPP is flawed. In particular, when applied to a partition pair, the result is not necessarily another partition pair or even well-defined. Moreover, when applied repeatedly to a transitive partition pair, convergence towards a unique fixed point is not guaranteed. Thereby we have shown that σ is not suitable for solving the GCPP. On the counterexample for the latter property, the algorithm of [8] that computes σ, produces a wrong result in which two simulation-equivalent states are put in different equivalence classes.

We have repaired this algorithm such that it correctly computes the solution of the GCPP. Apart from correcting the error that results from the flaws in the σ operator, we also corrected a minor mistake that caused premature termination of the algorithm on certain input. Our algorithm has the same space and time complexities as the original partitioning algorithm. We have proven its correctness using an auxiliary operator ρ of which we have shown that it solves the GCPP, though inefficiently. Finally, we have shown that no operator can be defined that captures the partition refinement performed in every iteration of our algorithm.

Another way to repair the algorithm of [8] may be to use the relation P^+ instead of P in REFINE$_{\text{GPP}}$. The thusly obtained algorithm would converge to a fixed point slightly slower than ours. More importantly, due to the cost of computing the transitive closure in each iteration, the time complexity would not match that of the original algorithm.

Acknowledgements. We would like to thank Raffaella Gentilini and Carla Piazza for answering some of our questions about their paper and providing us with their implementation of the algorithm.

References

[1] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation can't be traced.* Journal of the ACM 42(1), pp. 232–268.

[2] B. Bloom & R. Paige (1995): *Transformational design and implementation of a new efficient solution to the ready simulation problem.* Science of Computer Programming 24(3), pp. 189–220.

[3] D. Bustan & O. Grumberg (2003): *Simulation-based minimization.* ACM Transactions on Computational Logic 4(2), pp. 181–206.

[4] C. Courcoubetis, M.Y. Vardi, P. Wolper & M. Yannakakis (1990): *Memory efficient algorithms for the verification of temporal properties.* In Proc. 2nd Workshop on Computer-Aided Verification (CAV'90), LNCS 531, Springer, pp. 233–242.

[5] D. Dams, O. Grumberg & R. Gerth (1993): *Generation of reduced models for checking fragments of CTL.* In Proc. 5th Conference on Computer Aided Verification (CAV '93), LNCS 697, Springer, pp. 479–490.

[6] E.A. Emerson & J.Y. Halpern (1986): *"Sometimes" and "Not Never" revisited: On branching versus linear time temporal logic.* Journal of the ACM 33(1), pp. 151–178.

[7] S. Evangelista & J.-F. Pradat-Peyre (2005): *Memory efficient state space storage in explicit software model checking.* In Proc. 12th International SPIN Workshop on Model Checking Software, LNCS 3639, Springer, pp. 43–57.

[8] R. Gentilini, C. Piazza & A. Policriti (2003): *From bisimulation to simulation: Coarsest partition problems.* Journal of Automated Reasoning 31(1), pp. 73–103.

[9] R. Gentilini, C. Piazza & A. Policriti (2003): *From bisimulation to simulation: Coarsest partition problems.* RR 12-2003, Dep. of Computer Science, University of Udine, Italy.

[10] J.F. Groote & F.W. Vaandrager (1992): *Structured operational semantics and bisimulation as a congruence.* Information and Computation 100(2), pp. 202–260.

[11] M.R. Henzinger, T.A. Henzinger & P.W. Kopke (1995): *Computing simulations on finite and infinite graphs.* In 36th Annual Symposium on Foundations of Computer Science (FOCS'95), IEEE Computer Society Press, pp. 453–462.

[12] G.J. Holzmann (1988): *An improved protocol reachability analysis technique.* Software Practice and Experience 18(2), pp. 137–161.

[13] D. Kozen (1983): *Results on the propositional μ-calculus.* Theoretical Computer Science 27, pp. 333–354.

[14] A. Kucera & P. Jancar (2006): *Equivalence-checking on infinite-state systems: Techniques and results.* Theory and Practice of Logic Programming 6(3), pp. 227–264.

[15] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani & S. Bensalem (1995): *Property preserving abstractions for the verification of concurrent systems.* Formal Methods in System Design 6(1), pp. 11–44.

[16] D.M.R. Park (1981): *Concurrency and automata on infinite sequences.* In Proc. 5th GI-Conference on Theoretical Computer Science, LNCS 104, Springer, pp. 167–183.

[17] F. Ranzato & F. Tapparo (2007): *A new efficient simulation equivalence algorithm.* In Proc. 22nd Annual IEEE Symposium on Logic in Computer Science (LICS'07), IEEE Computer Society Press, pp. 171–180.

[18] L.J. Stockmeyer & A.R. Meyer (1973): *Word problems requiring exponential time.* In Proc. 5th Annual ACM Symposium on Theory of Computing (STOC'73), ACM, pp. 1–9.

[19] L. Tan & R. Cleaveland (2001): *Simulation revisited.* In Proc. 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01), LNCS 2031, Springer, pp. 480–495.