A Time and Space Efficient Simulation Algorithm

Francesco Ranzato Francesco Tapparo
Università di Padova, Italy

INTRODUCTION. A number of algorithms for computing the simulation preorder and equivalence are available [1–7]. The best simulation algorithms are those by Gentilini, Piazza and Policriti (GPP) [3] — subsequently corrected in [4] — and by Ranzato and Tapparo (RT) [6]. Let Σ denote the state space, $\rightarrow$ the transition relation and P_{sim} the partition of Σ induced by simulation equivalence. The algorithm GPP attains an optimal space bound of $O(|P_{\mathrm{sim}}|^2 + |\Sigma|\log|P_{\mathrm{sim}}|)$ — where optimal means that the space complexity is of the same order as the size of the output of the algorithm — while it runs in $O(|P_{\mathrm{sim}}|^2|{\rightarrow}|)$ time. The algorithm RT has the best time bound $O(|P_{\mathrm{sim}}||{\rightarrow}|)$ while it takes $O(|P_{\mathrm{sim}}||\Sigma|\log|\Sigma|)$ space. We present here a new time and space Efficient Simulation Algorithm (ESA) that runs in $O(|P_{\mathrm{sim}}||{\rightarrow}|\log|P_{\mathrm{sim}}|)$ time and $O(|P_{\mathrm{sim}}|^2\log|\Sigma| + |\Sigma|\log|P_{\mathrm{sim}}|)$ space. Thus, ESA reaches in practice the optimal space bound of GPP while significantly improving the time bound of GPP and approaching the best time bound of RT. Analogously to GPP and RT, ESA is a symbolic algorithm, meaning that it maintains and iteratively refines a partition P of the state space Σ that overapproximates the final simulation partition P_{sim} together with a preorder relation $\trianglelefteq$ over this partition P that instead overapproximates the final simulation preorder. Two main points allow to achieve the above complexity bounds: (1) ESA exploits a new efficient characterization of partition refiners, i.e., splitters for the current partition P and (2) ESA follows Hopcroft's "process the smaller half" principle when updating a crucial data structure involving P and $\trianglelefteq$ after a partition splitting. An experimental evaluation for comparing ESA to GPP and RT is ongoing.

BASIC SIMULATION ALGORITHM. If $S_1, S_2 \subseteq \Sigma$ then $S_1{\rightarrow}^{\exists}S_2$ means that there exist $x \in S_1$ and $y \in S_2$ such that $x{\rightarrow}y$. A *partition-relation pair* $\mathcal{P} = \langle P, \trianglelefteq\rangle$ [5], PR for short, is a state partition $P \in \mathrm{Part}(\Sigma)$ together with a binary relation $\trianglelefteq \subseteq P \times P$ between blocks of P. PRs allow to represent symbolically, i.e. through state partitions, a relation between states, in particular a preorder relation: A given preorder PR $\mathcal{P} = \langle P, \trianglelefteq\rangle$ induces the following preorder relation $\leq_{\mathcal{P}}$: $s \leq_{\mathcal{P}} t \;\Leftrightarrow\; P(s) \trianglelefteq P(t)$, so that for any block $B \in P$, $\mu_{\mathcal{P}}(B) \triangleq \cup\{C \in P \mid B \trianglelefteq C\}$ represents the set of states that simulate any state in B.

Theorem 1. Let $\mathcal{P} = \langle P, \trianglelefteq\rangle$ be a preorder PR. Then, $\mathcal{P}$ represents a simulation iff (i) if $B \trianglelefteq C$, $b \in B$ and $c \in C$ then $\ell(b) = \ell(c)$; (ii) if $B{\rightarrow}^{\exists}C$ and $B \trianglelefteq B'$ then $B'{\rightarrow}^{\exists}\mu_{\mathcal{P}}(C)$; (iii) for any $C \in P$, $P = \mathit{Split}(P, \mathrm{pre}(\mu_{\mathcal{P}}(C)))$.

```
ESA(PR ⟨$P_\ell$, id⟩)
  $Initialize()$;
  Bool RelationStable := ff, PartitionStable := ff;
  while ¬(RStable & PStable) do
    PartitionStable := ¬$RelationStabilize()$; RelationStable := tt;
    RelationStable := ¬$PartitionStabilize()$; PartitionStable := tt;

Bool $RelationStabilize()$
  Bool Removed := ff; Block $C$;
  while ($C$ := $FindRelationRefiner()$) ≠ null) do
    Removed := tt; $\trianglelefteq$ := RRefine($\trianglelefteq, C$);
  return Removed;

Bool $PartitionStabilize()$
  Bool Split := ff; Block $C$;
  while ($C$ := $FindPartitionRefiner()$) ≠ null) do
    Split := tt; $S := \mathrm{pre}(\mu_{\mathcal{P}}(C))$;
    $P := Split(P, S)$; $\trianglelefteq$ := PRefine($\trianglelefteq, S$);
  return Split;
```

Thus, given a block $C \in P$, (1) C is a *relation refiner* for $\mathcal{P}$ when there exist $B, D \in P$ such that $B{\rightarrow}^{\exists}C$, $B \trianglelefteq D$, but $D \not\rightarrow^{\exists}\mu_{\mathcal{P}}(C)$; (2) C is a *partition refiner* for $\mathcal{P}$ when $\mathrm{pre}(\mu_{\mathcal{P}}(C))$ splits the partition P. Then, it turns out that $\mathcal{P}$ is a simulation iff $\mathcal{P}$ satisfies condition (i) and has neither relation nor partition refiners. If C is a relation refiner and we refine $\trianglelefteq$ to the following relation:

$$\mathrm{RRefine}(\trianglelefteq, C) \triangleq \trianglelefteq \smallsetminus \{(B,E) \in P^2 \mid B{\rightarrow}^{\exists}C, B \trianglelefteq E, E \not\rightarrow^{\exists}\mu_{\mathcal{P}}(C)\}$$

then we are guaranteed that C is anymore a relation refiner. On the other hand, if C is a partition refiner then we first refine P to the partition $P' = \mathit{Split}(P, \mathrm{pre}(\mu_{\mathcal{P}}(C)))$ obtained by splitting P w.r.t. the splitter $S = \mathrm{pre}(\mu_{\mathcal{P}}(C))$, and subsequently $\trianglelefteq$ is modified to the following relation $\mathrm{PRefine}(\trianglelefteq)$ on P':

$$\mathrm{PRefine}(\trianglelefteq, S) \triangleq \{(D,E) \in P'^2 \mid \mathrm{parent}_P(D) \trianglelefteq \mathrm{parent}_P(E)\} \smallsetminus \{(B \cap S, B \smallsetminus S) \in P'^2 \mid B \in P \smallsetminus P'\}$$

namely, two blocks D and E of the new partition P' are related by $\mathrm{PRefine}(\trianglelefteq, S)$ if their parent blocks $\mathrm{parent}_P(D)$ and $\mathrm{parent}_P(E)$ in P were related by $\trianglelefteq$, except for the case $D = B \cap S$ and $E = B \smallsetminus S$ for any block $B \in P \smallsetminus P'$ that has been split by S. In this way, it turns out that C is anymore a partition refiner for $\mathcal{P}' = \langle P', \mathrm{PRefine}(\trianglelefteq, S)\rangle$, meaning that if $C' \in P'$ and $C' \subseteq C$ then C' is not a partition refiner for $\mathcal{P}'$.

This leads to design the basic algorithm ESA that iteratively refines the current PR $\mathcal{P}$ as described above until $\mathcal{P}$ becomes both relation and partition stable.

Correctness Theorem. If $\langle P, \trianglelefteq \rangle$ is the output PR of ESA then for any $s, t \in \Sigma$, $s \leq_{\text{sim}} t \;\Leftrightarrow\; P(s) \trianglelefteq P(t)$.

The following characterization of partition refiners is crucial for our efficient implementation of ESA.

Theorem 2. Let $\langle P, \trianglelefteq \rangle$ be a partial order PR. Then, $\langle P, \trianglelefteq \rangle$ has some partition refiner iff there exists $B, C \in P$ such that the following three conditions hold: (i) $B {\to}^{\exists} C$; (ii) for any $C' \triangleright C$, $B \not\to^{\exists} C'$; (iii) $B \not\subseteq \text{pre}(C)$.

IMPLEMENTATION.

• The states of any block B of the current partition P are consecutive in the list Σ, so that B is represented by two pointers begin and end. Also, any block B stores the list of blocks $C \in P$ such that $C {\to}^{\exists} B$ and its size in B.size. For any $C \in P$, $\mathit{Remove}(C)$ stores a set of blocks $E \in P$ such that if $\mathit{Remove}(C) = \varnothing$ then C is not a relation refiner and when instead $\mathit{Remove}(C) \neq \varnothing$ then any pair (B, E), with $B {\to}^{\exists} C$ and $E \not\to^{\exists} \mu_{\mathcal{P}}(C)$, is such that $E \in \mathit{Remove}(C)$. This leads to the algorithm *RelationStabilize*, where it is enough to scan all the blocks $B {\to}^{\exists} C$ and $E \in \mathit{Remove}(C)$ and when $B \trianglelefteq E = \textbf{tt}$ and $E \not\to^{\exists} \mu_{\mathcal{P}}(C)$ we must set $B \trianglelefteq E$ as **ff**. The remove sets satisfy the following property: if C_1 and C_2 are two blocks that are selected in two different iterations of the while-loop of *RelationStabilize*, possibly selected in two different calls of *RelationStabilize*, and $C_2 \subseteq C_1$ (possibly $C_1 = C_2$) then $(\cup \mathit{Remove}(C_1)) \cap (\cup \mathit{Remove}(C_2)) = \varnothing$. This property is crucial for obtaining that the overall time complexity $O(|P_{\text{sim}}||{\to}|)$ of *RelationStabilize*.

• The partition P is stored as a doubly linked list of blocks. The current relation $\trianglelefteq$ on P is stored as a resizable $|P| \times |P|$ boolean matrix so that insert operations for new blocks take amortized constant time.

• We store and maintain two resizable integer matrices BCount and Count indexed over $P \times P$:

$$\text{BCount}(B, C) \triangleq |\{(b, c) \mid b \in B,\ c \in C,\ b {\to} c\}|$$
$$\text{Count}(B, C) \triangleq \textstyle\sum_{E \trianglerighteq C} \text{BCount}(B, E).$$

Count allows to implement the test $B \not\to^{\exists} \text{pre}(\mu_{\mathcal{P}}(C))$ in constant time as $\text{Count}(B, C) = 0$, while BCount allows to implement in constant time conditions (i) and (ii) of Theorem 2 as $\text{BCount}(B, C) = \text{Count}(B, C)$. Theorem 2 is therefore implemented by the function *FindPartitionRefiner*().

• It turns out that all the data structures used in ESA can be quite easily maintained in $O(|P_{\text{sim}}||{\to}|)$ time, except for the integer matrix Count. This matrix Count needs to be updated in *RelationStabilize*() — and this is not a computational problem — and after a partition splitting. The simple algorithm *updateCount* updates the Count table for all the newly generated blocks by following Hopcroft's "process the smaller half" principle: if B is split into $B \cap S$ and $B \smallsetminus S$,

```
Bool $RelationStabilize()$
  Bool Removed := ff;
  while $\exists C \in P$ such that $Remove(C) \neq \varnothing$ do
    $Remove := Remove(C)$;  $Remove(C) := \varnothing$;
    forall $B \to^{\exists} C,\ E \in Remove$ do
      if ($B \trianglelefteq E$ & $\text{Count}(E, C) = 0$) then
        $B \trianglelefteq E$ := ff;  Removed := tt;
        forall $F \to^{\exists} E$ do
          $\text{Count}(F, B)$ -= $\text{BCount}(F, E)$;
          if ($\text{Count}(F, B) = 0$) then
            if ($Remove(B) = \varnothing$) then
              move $B$ at the end of $P$;
            $Remove(B)$.append($F$);
  return Removed;

Block $FindPartitionRefiner()$
  queue⟨Block⟩  $q := P$;
  while ($q \neq \varnothing$) do
    Block $B := q$.front();  list⟨Block⟩ $p := \{C \in P \mid B \to^{\exists} C\}$;
    forall $C \in p$ do
      if ($\text{BCount}(B, C) = \text{Count}(B, C)$) then return $C$;
    $q$.pop();
  return null;

$updateCount$(list⟨Block⟩ newBlocks)
  forall $B \in$ newBlocks do
    Block $X$, $Z$;
    if ($B$.size $\leq$ $B$.brother.size) then {$X := B$; $Z := B$.brother;}
    else  {$X := B$.brother;  $Z := B$;}
    forall $C \in P$ do
      $\text{Count}(Z, C) := \text{Count}(B, C)$;  $\text{Count}(X, C) := 0$;
    forall $x \in X$ do
      forall $y \to x$ do
        forall $C \in P$ such that $C \neq X$ & $C \trianglelefteq P(y)$ do
          Count($X$,$C$)++;  Count($Z$,$C$)−−;
    forall $D \in P$ do  $\text{Count}(D, B) := \text{Count}(D, B\text{.brother})$;
```

then we are able to update the rows $\text{Count}(B \cap S, \cdot)$ and $\text{Count}(B \smallsetminus S, \cdot)$ just by updating one of them, because for any C, $\text{Count}(B \cap S, C) + \text{Count}(B \smallsetminus S, C) = \text{Count}(B, C)$, so that it is enough to update the smaller in size between $B \cap S$ and $B \smallsetminus S$. Since the overall number of new blocks is in $O(|P_{\text{sim}}|)$ this guarantees that the overall update of the Count matrix can be done in $O(|P_{\text{sim}}||{\to}| \log |P_{\text{sim}}|)$ time.

Complexity Theorem. ESA runs in $O(|P_{\text{sim}}|^2 \log |\Sigma| + |\Sigma| \log |P_{\text{sim}}|)$ space and $O(|P_{\text{sim}}||{\to}| \log |P_{\text{sim}}|)$ time.

REFERENCES.

[1] D. Bustan and O. Grumberg. Simulation-based minimization. *ACM TOCL*, 4(2):181-204, 2003.

[2] S. Crafa, F. Ranzato and F. Tapparo. Saving space in a time efficient simulation algorithm. In *Proc. 9th ACSD*, 2009.

[3] R. Gentilini, C. Piazza and A. Policriti. From bisimulation to simulation: coarsest partition problems. *J. Automated Reasoning*, 31(1):73-103, 2003.

[4] R. van Glabbeek and B. Ploeger. Correcting a space-efficient simulation algorithm. In *Proc. 20th CAV*, LNCS 5123:517-529, 2008.

[5] M.R. Henzinger, T.A. Henzinger and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. 36th FOCS*, 453-462, 1995.

[6] F. Ranzato and F. Tapparo. A new efficient simulation equivalence algorithm. In *Proc. 22nd LICS*, 171-180, 2007.

[7] L. Tan and R. Cleaveland. Simulation revisited. In *Proc. 7th TACAS*, LNCS 2031:480-495, 2001.