Proofs of "An Automated Quantitative Information Flow Analysis for Concurrent Programs" presented in QEST-2022

Khayyam Salehi[1][0000−0002−3379−798X], Ali A. Noroozi[2][0000−0003−1173−079X], Sepehr Amir-Mohammadian[3][0000−0002−2301−4283], and Mohammadsadegh Mohagheghi[4][0000−0001−8059−3691]

[1] Department of Computer Science, Shahrekord University, Shahrekord, Iran
kh.salehi@sku.ac.ir
[2] Department of Computer Science, University of Tabriz, Tabriz, Iran
noroozi@tabrizu.ac.ir
[3] Department of Computer Science, University of the Pacific, Stockton, CA, USA
samirmohammadian@pacific.edu
[4] Department of Computer Science, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran
mohagheghi@vru.ac.ir

A Proofs

Theorem 1. *Back-bisimulation is an equivalence relation.*

Proof. Let $\mathcal{M}_\delta^{\mathtt{P}}$ be an MC. We show reflexivity, symmetry, and transitivity of the relation $\sim_b$.

- Reflexivity: It is obvious that $s \sim_b s$ for all states $s \in S$.
- Symmetry: Assume that $s_1 \sim_b s_2$. We should show that $s_2 \sim_b s_1$. Clearly, condition (1) holds. By symmetry of conditions (2) and (3), we immediately conclude that $s_2 \sim_b s_1$.
- Transitivity: Let $s_1 \sim_b s_2$ and $s_2 \sim_b s_3$. We should show that $s_1 \sim_b s_3$.
  (1) As $s_1 \sim_b s_2$ and $s_2 \sim_b s_3$, it follows that $V(s_1) = V(s_2) = V(s_3)$.
  (2) Assume $s_1 \sim_b s_3$. Since $s_1 \sim_b s_2$, it follows that if $s_1' \in Pre(s_1)$ then $s_1' \sim_b s_2'$ for some $s_2' \in Pre(s_2)$. Since $s_2 \sim_b s_3$, we have $s_2' \sim_b s_3'$ for some $s_3' \in Pre(s_3)$. Hence, $s_1' \sim_b s_3'$.
  (3) Similar to the proof for item (2).

Theorem 2. *Let $\mathcal{M}_\delta^{\mathtt{P}}$ be an $MC_{\mathtt{n}}$. For all paths $\sigma_1, \sigma_2 \in Paths(\mathcal{M}_\delta^{\mathtt{P}})$ with $\sigma_1 = s_{0,1}s_{1,1}\ldots s_{n-1,1}(s_{n,1})^\omega$, $\sigma_2 = s_{0,2}s_{1,2}\ldots s_{n-1,2}(s_{n,2})^\omega$, and $n \geq 0$ it holds that $s_{n,1} \sim_b s_{n,2}$ iff $trace(\sigma_1) = trace(\sigma_2)$.*

Proof. The proof is carried out in two steps.

$\Rightarrow$: Assume $s_{n,1} \sim_b s_{n,2}$. We show that $trace(\sigma_1) = trace(\sigma_2)$. From $s_{n,1} \sim_b s_{n,2}$, it immediately follows that $V(s_{n,1}) = V(s_{n,2})$ and $s_{n-1,1} \sim_b s_{n-1,2}$. The latter yields $V(s_{n-1,1}) = V(s_{n-1,2})$ and $s_{n-2,1} \sim_b s_{n-2,2}$. This inductive label-equality can be continued until the initial states: $V(s_{0,1}) = V(s_{0,2})$. Therefore, for all $0 \leq i \leq n$, $V(s_{i,1}) = V(s_{i,2})$, which yields $trace(\sigma_1) = trace(\sigma_2)$.

$\Leftarrow$: Assume $trace(\sigma_1) = trace(\sigma_2)$. We show that $s_{n,1} \sim_b s_{n,2}$. From $trace(\sigma_1) = trace(\sigma_2)$, it follows that $V(s_{i,1}) = V(s_{i,2})$ for $0 \leq i \leq n$. States $s_{n,1}$ and $s_{n,2}$ have intersecting pre-labels:

$$V(s_{n-1,1}) = V(s_{n-1,2}) \in PreLabels(s_{n,1}) \cap PreLabels(s_{n,2}).$$

Since $\mathcal{M}^{\mathfrak{p}}_{\delta}$ is an $\mathrm{MC}_{\mathfrak{n}}$, the states $s_{n,1}$ and $s_{n,2}$ are not pseudoback-bisimilar. From the definition of pseudoback-bisimulation (Definition 9) and considering that $V(s_{n,1}) = V(s_{n,2})$, $level(s_{n,1}) = level(s_{n,2}) = n$, and $PreLabels(s_1) \cap PreLabels(s_2) \neq \varnothing$, it follows that $sig_{\sim_b}(s_1) = sig_{\sim_b}(s_2)$. This yields $s_{n,1} \sim_b s_{n,2}$.

Theorem 3. *Let $\mathcal{M}^{\mathfrak{p}}_{\delta}$ be an $MC_{\mathfrak{n}}$. For all paths $\sigma_1, \sigma_2 \in Paths(\mathcal{M}^{\mathfrak{p}}_{\delta})$ with $\sigma_1 = s_{0,1}s_{1,1}\ldots s_{n-1,1}(s_{n,1})^{\omega}$, $\sigma_2 = s_{0,2}s_{1,2}\ldots s_{m-1,2}(s_{m,2})^{\omega}$, $n, m > 0$, and $0 \leq i < min(n,m)$ it holds that $s_{i,1} \sim_b s_{i,2}$ iff $trace_{\ll i}(\sigma_1) = trace_{\ll i}(\sigma_2)$.*

Proof. Proof is similar to the proof of theorem 2, and is omitted to avoid repetition.

Theorem 4. *Algorithm 1 always terminates and correctly computes the back-bisimulation quotient space $S/\sim_b$.*

Proof. Termination of Algorithm 1 is proven by Lemma 1. The correctness of the refinement operator is proven by Lemma 2. It shows that successive refinements, starting with partition Π_0, yield a series of partitions $\Pi_0, \Pi_1, \Pi_2, \ldots$. These partitions become increasingly finer and all are coarser than $S/\sim_b$. For partitions Π_1 and Π_2 of S, Π_1 is called finer than Π_2, or Π_2 is called coarser than Π_1, if:

$$\forall B_1 \in \Pi_1 \; \exists B_2 \in \Pi_2. \;\; B_1 \subseteq B_2.$$

Lemma 3 proves that $S/\sim_b$ is the coarsest partition for S. Thus, successive refinements of Algorithm 1 yield $S/\sim_b$. This shows that Algorithm 1 correctly computes $S/\sim_b$.

Lemma 1. *Algorithm 1 always terminates.*

Proof. Due to the definition of $Refine_b(\Pi, C)$, the partition Π is finer than Π_{old}, i.e. $\forall B_1 \in \Pi \; \exists B_2 \in \Pi_{old}, B_1 \subseteq B_2$. According to finiteness of S, a partition Π with $\Pi = \Pi_{old}$ (line 6 of the algorithm) is reached after at most $|S|$ iterations. In other words, after $|S|$ refinements, any block in Π is a singleton, and the algorithm always terminates.

Lemma 2. *Let Π be a partition of S, which is finer than Π_0 and coarser than $S/\sim_b$ and C be a superblock of Π. Then:*
(a) $Refine(\Pi, C)$ is finer than Π.
(b) $Refine(\Pi, C)$ is coarser than $S/\sim_b$.

Proof.

(a) This follows directly from the definition of $Refine$ (Definition 10), since every block $B \in \Pi$ is either contained in $Refine(\Pi, C)$ or is decomposed into $B \cap Post(C)$ and $B \setminus Post(C)$.

(b) To prove that $Refine(\Pi, C)$ is coarser than $S/\sim_b$, we need to prove that each block B in $S/\sim_b$ is contained in a block of $Refine(\Pi, C)$. Since Π is coarser than $S/\sim_b$ (part (a)), there exists a block $B' \in \Pi$ with $B \subseteq B'$. B' is of the form $B' = B \cup D$ where D is a (possibly empty) superblock of $S/\sim_b$. If $B' \in Refine(\Pi, C)$, the $B \subseteq B' \subseteq Refine(\Pi, C)$. Otherwise, i.e., if $B' \notin Refine(\Pi, C)$, then due to the definition of $Refine(\Pi, C)$ (Definition 10), B' is decomposed into the subblocks $B' \cap Post(C)$ and $B' \setminus Post(C)$. It remains to show that B is included in one of these two new subblocks. Condition (ii) of the previous lemma implies that either $B \cap Post(C) = \varnothing$ ($B \setminus Post(C) = B$) or $B \setminus Post(C) = \varnothing$ ($B \cap Post(C) = B$). Since $B' = B \cup D$, B is either contained in block
- $B' \setminus Post(C) = (B \setminus Post(C)) \cup (D \setminus Post(C))$
- or in $B' \cap Post(C) = (B \cap Post(C)) \cup (D \cap Post(C))$.

Lemma 3. *The back-bisimulation quotient space $S/\sim_b$ is the coarsest partition Π for S such that:*
(i) Π is finer than Π_0.
(ii) for all $B, C \in \Pi$: $B \cap Post(C) = \varnothing$ or $B \subseteq Post(C)$.
Remember that $Post(C) = \{s \in S | Pre(s) \cap C \neq \varnothing\}$ describes the set of states in S, which have at least one predecessor in C.

Proof. Let Π be a partition of S and $\mathcal{R}_\Pi$ the equivalence relation on S induced by Π. The proof is carried out in two steps. The first step is to prove that $\mathcal{R}_\Pi$ is a back-bisimulation if and only if the conditions (i) and (ii) are satisfied. The last step is to show that $S/\sim_b$ is the coarsest partition satisfying (i) and (ii).

$\Leftarrow$: *Assume that Π satisfies (i) and (ii). We prove that $\mathcal{R}_\Pi$ is a back-bisimulation. Let $(s_1, s_2) \in \mathcal{R}_\Pi$ and $B = [s_1]_\Pi = [s_2]_\Pi$.*

1. *Since Π is finer than Π_0 (condition (i)), there exists a block B' of Π_0 containing B. Thus, $s_1, s_2 \in B \subseteq B' \in \Pi_0$. Since the public variables in each block of Π_o is the same (line 3 of the algorithm), we have $L(s_1) = L(s_2)$.*
2. *Let s_1' be one of the predecessors of s_1, i.e. $s_1' \in Pre(s_1)$ and C be an equivalence class of s_1', i.e. $C = [s_1']_\Pi$. Then, $s_1 \in B \cap Post(C)$. By condition (ii), we obtain $B \subseteq Post(C)$. Hence, $s_2 \in Post(C)$. So, there exists a state $s_2' \in Pre(s_2) \cap C$. Because $s_2' \in C = [s_1']_\Pi$, it results that $(s_1', s_2') \in R_\Pi$.*

$\Rightarrow$: *Assume $\mathcal{R}_\Pi$ is a back-bisimulation. It remains to show that the conditions (i) and (ii) are satisfied.*

(i) By contradiction. Assume that Π is not finer than Π_0. Then, there exist a block $B \in \Pi$ and states $s_1, s_2 \in B$ with $[s_1]_{\Pi_0} \neq [s_2]_{\Pi_0}$. Then, according to

the definition of Π_0, $L(s_1) \neq L(s_2)$. Hence, $\mathcal{R}_\Pi$ is not a back-bisimulation relation. Contradiction.

(ii) This is proved in two steps. First, we have to prove that condition (ii) is satisfied when B, C are blocks of Π. We assume that $B \cap Post(C) \neq \varnothing$ and show that $B \subseteq Post(C)$. Since $B \cap Post(C) \neq \varnothing$, there exist a state $s_1 \in B$ with $Pre(s_1) \cap C \neq \varnothing$, that is there exist a predecessor of s_1 in C. Let $s_1' \in Pre(s_1) \cap C$ and s_2 be an arbitrary state of B. We deduce that $s_2 \in Post(C)$. Since $s_1, s_2 \in B$, we get that $(s_1, s_2) \in \mathcal{R}_\Pi$. According to $s_1' \longrightarrow s_1$, there exists a transition $s_2' \longrightarrow s_2$ with $(s_1', s_2') \in \mathcal{R}_\Pi$. Since $s_1' \in C$ we have $s_2' \in C$. Thus, $s_2' \in Pre(s_2) \cap C$ and $s_2 \in Post(C)$.
In the last step, we prove that (ii) is satisfied for block B and superblock C of Π. Assume Π satisfies (ii). We show that $B \cap Post(C) = \varnothing$ or $B \subseteq Post(C)$. Let $B \in \Pi$ and C a superblock, i.e., C is of the form $C = C_1 \cup \cdots \cup C_l$ for blocks $C_1, \ldots, C_l$ of Π. Assume that $B \cap Post(C) \neq \varnothing$. We have to prove that $B \subseteq Post(C)$. Since $B \cap Post(C) \neq \varnothing$, there is an index $i \in \{1, \ldots l\}$ with $B \cap Post(C_i) \neq \varnothing$. So by the first step, $B \subseteq Post(C_i) \subseteq Post(C)$.

It remains to show that the back-bisimulation partition $\Pi = S/\sim_b$ is the coarsest partition of S; This immediately follows from the definition of $\sim_b$.

Theorem 5. *The time complexity of Algorithm 1 is $O(|S|.|E|)$, where E denotes the set of transitions of $\mathcal{M}_\delta^{\mathsf{p}}$.*

Proof. In order to compute the initial partition, a *hash map* could be used. Hash map is a data structure for mapping *keys* to *values*. Here, keys are possible values of `l` and values are blocks of states. The time complexity of inserting a key-value pair to the hash map is $O(1)$ in average and $O\big(min\big(|Val_l|, |S|\big)\big)$ in worst case. This yields the overall time complexity of $O\Big(|S|.min\big(|Val_l|, |S|\big)\Big)$ for computing the initial partition.

In refining each partition, $Refine_b(\Pi, \mathcal{C})$ causes the cost $O(|Post(s)|+1)$ for each state $s \in S$. The summand 1 reflects the case $Post(s) = \varnothing$. The outermost iteration is traversed maximally $|S|$ times. Thus, the overall cost of successive partition refinements is

$$O\Bigg(|S|.\sum_{s\in S}(|Post(s)|+1)\Bigg) =$$

$$O\Bigg(|S|.\Big(\sum_{s\in S}|Post(s)| + |S|\Big)\Bigg).$$

Let $E = \sum_{s\in S}|Post(s)|$ denote the number of transitions of $\mathcal{M}_\delta^{\mathsf{p}}$. Assuming $E \geq |S|$, the latter complexity can be simplified to $O(|S|.E)$.

Finally, the overall time complexity of Algorithm 1 is computed as

$$O\Big(|S|.min\big(|Val_l|, |S|\big) + |S|.E\Big) =$$
$$O\big(|S|.E\big).$$