Deriving Bisimulation Relations from Path Extension Based Equivalence Checkers

Kunal Banerjee, Chittaranjan Mandal, Dipankar Sarkar
Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur
{kunalb,chitta,ds}@cse.iitkgp.ernet.in

1. Problem statement

Constructing bisimulation relations between programs as a means of translation validation has been an active field of study [3, 4]. The problem is in general undecidable. Currently available mechanisms suffer from drawbacks such as non-termination and significant restriction on the structures of programs to be checked. We have developed path extension based equivalence checking [1, 2] as an alternative translation validation technique to alleviate these drawbacks. In this work, path extension based equivalence checking of programs (flowcharts) are leveraged to establish a bisimulation relation between a program and its translated version by constructing the relation from the outputs of the equivalence checker.

2. The FSMD model and (bi)simulation relation between FSMD models

The FSMD model for equivalence checking is described in brief; a detailed description can be found in [1]. The FSMD model is formally defined as an ordered tuple $\langle Q, q_0, I, V, O, \tau : Q \times 2^S \rightarrow Q,\ h\ :\ Q \times 2^S \rightarrow U \rangle$, where Q is the finite set of control states, q_0 is the reset state, I is the set of input variables which are never changed in course of a computation of the model, V is the set of storage variables, O is the set of output variables, τ is the state transition function, h is the update function of the output and the storage variables, U represents a set of storage and output assignments and $\mathcal{S}$ represents a set of status signals as relations between two arithmetic expressions over the members of I and V.

Let $\overline{v}$ denote a vector (an ordered tuple) of variables of I and V assuming values from the domain $D_{\overline{v}} = \Pi_{v \in I \cup V} D_v$, where D_v is the domain of the variable v and Π stands for Cartesian product of sets. Let $q(\overline{v}), q \in Q$, denote the set of values assumed by the variables in the state q. Each element of $q(\overline{v})$ is called a data state at q and is denoted as σ_q. Let $\Sigma = \bigcup_{q \in Q} \sigma_q$ be the set of all the data states.

A *path* α in an FSMD model is a finite sequence of states where the first and the last states are cut-points, there are no intermediary cut-points and at most the last state may be non-distinct (in case of a loop) and any two consecutive states in the sequence are in τ. The initial (start) and the final control states of a path α are denoted as α^s and α^f, respectively. A path α is characterized by an ordered pair $\langle R_\alpha, r_\alpha \rangle$, where R_α is the condition of execution of α satisfying the property that $\forall \sigma_{\alpha^s} \in \Sigma$, $R_\alpha(\sigma_{\alpha^s})$ implies that the path α is executed if control reaches the state α^s; the second member $r_\alpha = \langle \mathsf{s}_\alpha, \theta_\alpha \rangle$, where s_α is the functional transformation of the path α, i.e., $\sigma_{\alpha^f} = \mathsf{s}_\alpha(\sigma_{\alpha^s})$ and θ_α represents the output list of data values of some variables produced along the path α. Two paths α and β are said to be computationally equivalent, denoted as $\alpha \simeq \beta$, iff $R_\alpha \equiv R_\beta$ and $r_\alpha = r_\beta$.

It is worth noting that if two behaviors are to be computationally equivalent, then their outputs must match. Since equivalence of θ_α hinges upon the equivalence of s_α, the rest of the paper focuses on computation of s_α and the computation of θ_α has deliberately been omitted for the sake of brevity.

Any computation μ of an FSMD M can be looked upon as a computation along some concatenated path $[\alpha_1 \alpha_2 \alpha_3 ... \alpha_k]$ of M such that, for $1 \leq i < k$, α_i terminates in the initial state of the path α_{i+1}, the path α_1 emanates from and the path α_k terminates in the reset state q_0 of M; α_i's may not all be distinct. Hence, we have the following definition.

Definition 1 (Path cover of an FSMD)**.** *A finite set of paths $P = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k\}$ is said to be a path cover of an FSMD M if any computation μ of M can be looked upon as a concatenation of paths from P.*

The set of all paths from a cut-point to another cut-point without having any intermediary cut-point is a path cover of the FSMD. In course of establishing equivalence of paths of a path cover, a path based equivalence checker produces a natural correspondence between the control states from two FSMDs as defined below.

Definition 2 (Corresponding States)**.** *Let $M_1 = \langle Q_1, q_{1,0}, I, V_1, O, \tau_1, h_1 \rangle$ and $M_2 = \langle Q_2, q_{2,0}, I, V_2, O, \tau_2, h_2 \rangle$ be two FSMDs having identical input and output sets, I and O, respectively.*
1) The respective reset states $q_{1,0}$ and $q_{2,0}$ are corresponding states and a non-reset state does not have correspondence with a reset state.
2) If $q_{1,i} \in Q_1$ and $q_{2,j} \in Q_2$ are corresponding states and there exist $q_{1,k} \in Q_1$ and $q_{2,l} \in Q_2$ such that, for some path α from $q_{1,i}$ to $q_{1,k}$ in M_1, there exists a path β from $q_{2,j}$ to $q_{2,l}$ in M_2 such that $\alpha \simeq \beta$, then $q_{1,k}$ and $q_{2,l}$ are corresponding states.

The relation of corresponding states is denoted as $\delta \subseteq Q_1 \times Q_2$. The following theorem can be concluded from the above discussion.

Theorem 1. *An FSMD M_1 is contained in another FSMD M_2 ($M_1 \sqsubseteq M_2$), if there exists a finite path cover $P_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_{l_1}\}$ of M_1 for which there exists a set $P_2 = \{\beta_1, \beta_2, \ldots, \beta_{l_2}\}$ of paths of M_2 such that for any corresponding state pair $\langle q_{1,i}, q_{2,j} \rangle$, for any path $\alpha_m \in P_1$ emanating from $q_{1,i}$, there exists a path $\beta_n \in P_2$ emanating from $q_{2,j}$ such that $\alpha_m \simeq \beta_n$.*

We use the notation $q_i \twoheadrightarrow q_j$ to represent a path from the state q_i to the state q_j. For the subsequent discussion, we suffix the (set of) data states in some control state of the machine $M_i, i \in \{1, 2\}$, in a natural way. Similarly, we rename the variables occurring in FSMD M_i by adding the suffix i. Thus, for machine $M_i, i \in \{1, 2\}$, $\sigma_i \in \Sigma_i$ represents a data state of M_i and $\sigma_{i,j} \in q_{i,j}(\overline{v_i})$ represents a data state at the control state $q_{i,j}$. Let the predicate $\phi(\sigma_1, \sigma_2)$ indicate that ϕ is satisfied by the data states σ_1 of M_1 and σ_2 of M_2. A verification relation between two FSMDs M_1 and M_2 is a set of triples $\langle q_{1,i}, q_{2,j}, \phi_{i,j} \rangle$, where $q_{1,i} \in Q_1$, $q_{2,j} \in Q_2$, $\phi_{i,j} \subseteq q_{1,i}(\overline{v_1}) \times q_{2,j}(\overline{v_2})$. Based on the above discussions, a simulation relation can be defined as follows (in accordance with the definition of simulation relation given in [3]).

Definition 3 (Simulation Relation)**.** *A simulation relation S for two FSMDs $M_1 = \langle Q_1, q_{1,0}, I, V_1, O, \tau_1, h_1 \rangle$ and $M_2 = \langle Q_2, q_{2,0}, I, V_2, O, \tau_2, h_2 \rangle$ is a verification relation which satisfies the following two clauses:*
1. $S(q_{1,0}, q_{2,0},\ true)$, and
2. $\forall q_{1,i}, q_{1,k} \in Q_1$, $\sigma_{1,i} \in \Sigma_1$, $q_{1,i} \twoheadrightarrow q_{1,k} \in P_1$, $q_{2,j} \in Q_2$, $\sigma_{2,j} \in \Sigma_2 \left[R_{q_{1,i} \twoheadrightarrow q_{1,k}}(\sigma_{1,i}) \wedge \phi_{i,j}(\sigma_{1,i}, \sigma_{2,j}) \wedge S(q_{1,i}, q_{2,j}, \phi_{i,j}) \right.$
$\Rightarrow \quad \exists \sigma_{1,k} \in \Sigma_1$, $q_{2,l} \in Q_2$, $\sigma_{2,l} \in \Sigma_2$, $q_{2,j} \twoheadrightarrow q_{2,l} \in P_2$

$\{\ R_{q_{2,j} \twoheadrightarrow q_{2,l}}(\sigma_{2,j}) \wedge \{(q_{1,i} \twoheadrightarrow q_{1,k}) \simeq (q_{2,j} \twoheadrightarrow q_{2,l})\} \wedge$
$\phi_{k,l}(\sigma_{1,k}, \sigma_{2,l})\ \wedge S(q_{1,k}, q_{2,l}, \phi_{k,l})\ \}\ ]$.

Intuitively, the two clauses mentioned above state that: 1) the reset state of FSMD M_1 must be related to the reset state of FSMD M_2 for all data states, and 2) if two states of M_1 and M_2 with their respective data states are related and M_1 can proceed along a path, α say, then M_2 must also be able to proceed along a path, β say, such that $\alpha \simeq \beta$ and the end states of the two paths must also be related. We now define a *bisimulation relation* using the definition of simulation relation.

Definition 4 (Bisimulation Relation)**.** *A verification relation B for two FSMDs M_1 and M_2 is a bisimulation relation between them iff B is a simulation relation for M_1, M_2 and $B^{-1} = \{(q_{2,j}, q_{1,i}, \phi)\ |\ B(q_{1,i}, q_{2,j}, \phi)\}$ is a simulation relation for M_2, M_1.*

3. Deriving simulation relation from the output of a path extension based equivalence checker

Algorithm 1 *deriveSimReln (FSMD M_1, FSMD M_2, Set δ)*

Inputs: Two FSMDs M_1 and M_2, and the set δ of corresponding state pairs obtained from M_1 and M_2 by the path extension based equivalence checker.

Outputs: A relation $S = \{\langle q_{1,k}, q_{2,l}, \phi_{k,l}\rangle | \langle q_{1,k}, q_{2,l}\rangle \in \delta\}$.

1: Let the relation S be empty.
2: Perform live variable analyses on M_1 and M_2 and compute the set of live variables for each of the states that appears as a member of the state pairs in δ.
3: Rename each $v_j \in V_i$ as $v_{i,j}, i \in \{1, 2\}$.
4: For the pair $\langle q_{1,0}, q_{2,0}\rangle$ in δ, let $\phi_{0,0}$ be $true$ and $S \leftarrow S \cup \{\langle q_{1,0}, q_{2,0}, \phi_{0,0}\rangle\}$.
5: For each of the other state pairs $\langle q_{1,i}, q_{2,j}\rangle$ in δ, let $\phi_{i,j}$ be $v_{1,k_1} = v_{2,k_1} \wedge \ldots \wedge v_{1,k_n} = v_{2,k_n}$, where $v_{k_1}, \ldots, v_{k_n}$ are the common variables live at $q_{1,i}$ and $q_{2,j}$; $S \leftarrow S \cup \{\langle q_{1,i}, q_{2,j}, \phi_{i,j}\rangle\}$.
6: **return** S.

A member of the form $\langle q_{1,i}, q_{2,j}, \phi_{i,j}\rangle$ in a simulation relation indicates that the data states at $q_{1,i}$ and $q_{2,j}$ satisfy the predicate $\phi_{i,j}$. The predicate $\phi_{i,j}$ involves the variables appearing in the two FSMDs as free variables. For path based equivalence checkers, this formula is $true$ for the pair of reset states and for any other pair of corresponding states $\langle q_{1,i}, q_{2,j}\rangle$, it is of the form $v_{1,k_1} = v_{2,k_1} \wedge \ldots \wedge v_{1,k_n} = v_{2,k_n}$, where $v_{k_1}, \ldots, v_{k_n}$ are the common variables live at $q_{1,i}$ and $q_{2,j}$, which is precisely the criterion that must be satisfied for the two paths to be declared equivalent by a path based equivalence checker. Note that the variables that appear in both the FSMDs M_1 and M_2 are termed as *common variables*; other variables (that appear in either of the FSMDs but not both) are termed as *uncommon variables*. It is also to be noted that presence of *live* uncommon variables always leads to path extension; hence, at the corresponding state pairs, there is no live uncommon variable – consequently, the uncommon variables do not appear in the $\phi_{i,j}$'s.

The following theorem captures the fact that path based equivalence checking leads to a simulation relation between two FSMDs. However, before proceeding with the theorem and its corresponding proof, some concepts need to be introduced. We use the notation $\overline{v_c}$ for representing the name vector comprising the common variables in $V_1 \cap V_2$; the vector $\overline{v_c}$ assumes values from the domain $(\Sigma_1 \cup \Sigma_2)|_{\overline{v_c}}$. Let $\overline{v_{1c}}$ and $\overline{v_{2c}}$ represent the name vectors over the common variables after renaming the components of $\overline{v_c}$ by respectively adding suffix 1 for FSMD M_1 and suffix 2 for FSMD M_2. Thus, $\overline{v_{1c}}$ ($\overline{v_{2c}}$) assumes values from the domain $\Sigma_1|_{\overline{v_c}}$ ($\Sigma_2|_{\overline{v_c}}$). We additionally use the symbol $L_{i,j}(\overline{v_i}), i \in \{1, 2\}$ to denote a vector containing only those variables from $\overline{v_i}$ that are live at state $q_{i,j}$; the operation $\{\overline{e}/\overline{v}\}$ is called substitution. Let $\sigma_{1,i} \in q_{1,i}(\overline{v_1})$ and $\sigma_{2,j} \in q_{2,j}(\overline{v_2})$; then $L_{1,i}(\sigma_{1,i})$ represents the values assumed by the live variables corresponding to the data state $\sigma_{1,i}$ at the control state $q_{1,i}$ in M_1; $L_{2,j}(\sigma_{2,j})$ is defined similarly.

Theorem 2. *If a path based equivalence checker declares an FSMD M_1 to be contained in another FSMD M_2, then there is a simulation relation between M_1 and M_2 corresponding to the state correspondence relation δ produced by the equivalence checker. Symbolically,*
$\forall \langle q_{1,i}, q_{2,j}\rangle \in Q_1 \times Q_2,\ \delta(q_{1,i}, q_{2,j}) \Rightarrow \exists \phi_{i,j}\ S(q_{1,i}, q_{2,j}, \phi_{i,j})$.

Proof: Construction of $\phi_{i,j}$'s and the simulation relation S: Algorithm 1 shows the steps to obtain the simulation relation from the output of a path based equivalence checker. The notation $L_{1,i}(\overline{v_{1c}}) = L_{2,j}(\overline{v_{2c}})$ implies that all the common variables that are live at $q_{1,i}$ in M_1 and $q_{2,j}$ in M_2 assume the same values at these states; symbolically, $\forall \sigma_{1,i} \in \Sigma_1, \sigma_{2,j} \in \Sigma_2,\ L_{1,i}(\overline{v_{1c}})\{\sigma_{1,i}|_{\overline{v_{1c}}/\overline{v_{1c}}}\} = L_{2,j}(\overline{v_{2c}})\{\sigma_{2,j}|_{\overline{v_{2c}}/\overline{v_{2c}}}\}$. (Algorithm 1 depicts this equality more elaborately as a conjunction of equalities of the corresponding components of the vectors $L_{1,i}(\overline{v_{1c}})$ and $L_{2,j}(\overline{v_{2c}})$.) Now, we prove that the relation S constructed in Algorithm 1 is indeed a simulation relation, i.e., the relation S conforms with Definition 3. Consider any $\langle q_{1,i}, q_{2,j}\rangle \in Q_1 \times Q_2$ such that $\delta(q_{1,i}, q_{2,j})$ holds. Hence, from steps 4 and 5 of Algorithm 1, $\langle q_{1,i}, q_{2,j}, \phi_{i,j}\rangle \in S$, where

$$\phi_{i,j} \triangleq true \text{ if } q_{1,i} = q_{1,0},\ q_{2,j} = q_{2,0} \text{ and}$$
$$\triangleq L_{1,i}(\overline{v_{1c}}) = L_{2,j}(\overline{v_{2c}}),\ \text{otherwise.}$$

Obviously, clause 1 of Definition 3 holds by step 4 of Algorithm 1. Conformance with clause 2 is as follows. From the prefix quantification of clause 2, we know $q_{1,i} \twoheadrightarrow q_{1,k}\ (= \alpha, \text{say}) \in P_1$ and from conjunct 1 of the antecedent it follows that $R_\alpha(\sigma_{1,i})$ holds. The path based equivalence checker ensures that $\exists \beta = q_{2,j} \twoheadrightarrow q_{2,l} \in P_2$ such that $R_\beta(\sigma_{2,j})$ holds and $\alpha \simeq \beta$; thus, conjuncts 1 and 2 of the consequent and $\delta(q_{1,k}, q_{2,l})$ hold; the inference $\delta(q_{1,k}, q_{2,l})$ follows from Definition 2. Now let us prove the third conjunct, i.e., $\phi_{k,l}$ holds for some $\sigma_{1,k} \in q_{1,k}(\overline{v_{1c}})$ and $\sigma_{2,l} \in q_{2,l}(\overline{v_{2c}})$. From step 5 of Algorithm 1,

$$\phi_{k,l} \triangleq L_{1,k}(\overline{v_{1c}}) = L_{2,l}(\overline{v_{2c}})$$
$$\therefore \phi_{k,l}(\sigma_{1,k}, \sigma_{2,l}) \triangleq L_{1,k}(\sigma_{1,k}|_{\overline{v_{1c}}/\overline{v_{1c}}}) = L_{2,l}(\sigma_{2,l}|_{\overline{v_{2c}}/\overline{v_{2c}}})$$
$$\triangleq L_{1,k}(\sigma_{1,k}|_{\overline{v_{1c}}}) = L_{2,l}(\sigma_{2,l}|_{\overline{v_{2c}}})$$

Let $\sigma_{1,k}$ be $\mathsf{s}_\alpha(\sigma_{1,i})$ and $\sigma_{2,l}$ be $\mathsf{s}_\beta(\sigma_{2,j})$. The equivalence checker has identified $\alpha \simeq \beta$ by finding that the live common variables are equal at $\alpha^f = q_{1,k}$ and $\beta^f = q_{2,l}$. Thus, if α is executed with $\sigma_{1,i} \in q_{1,i}(\overline{v_1})$ and β is executed with $\sigma_{2,j} \in q_{2,j}(\overline{v_2})$, then
$L_{1,k}(\mathsf{s}_\alpha(\sigma_{1,i})|_{\overline{v_{1c}}}) = L_{2,l}(\mathsf{s}_\beta(\sigma_{2,j})|_{\overline{v_{2c}}})$ holds;
$\Rightarrow L_{1,k}(\sigma_{1,k}|_{\overline{v_{1c}}}) = L_{2,l}(\sigma_{2,l}|_{\overline{v_{2c}}})$ holds;
$\Rightarrow \phi_{k,l}(\sigma_{1,k}, \sigma_{2,l})$ holds.
Again from step 5 of Algorithm 1, $\delta(q_{1,k}, q_{2,l}) \Rightarrow S(q_{1,k}, q_{2,l}, \phi_{k,l})$, conjunct 4 of the consequent, where $\phi_{k,l} \triangleq L_{1,i}(\overline{v_{1c}}) = L_{2,j}(\overline{v_{2c}})$, holds. ■

Note that the path extension based equivalence checker ensures $M_1 \sqsubseteq M_2$ first and then $M_2 \sqsubseteq M_1$. In the second step, it does not have to change the set δ of corresponding state pairs constructed during the first step. Hence, the simulation relation S obtained from Algorithm 1 will be a bisimulation relation too by Definition 4.

References

[1] C. Karfa, D. Sarkar, C. Mandal, and P. Kumar. An equivalence-checking method for scheduling verification in high-level synthesis. *IEEE Trans on CAD of ICS*, 27:556–569, 2008.

[2] C. Karfa, C. Mandal, and D. Sarkar. Formal verification of code motion techniques using data-flow-driven equivalence checking. *ACM Trans. Design Autom. Electr. Syst.*, 17(3):30, 2012.

[3] S. Kundu, S. Lerner, and R. Gupta. Translation validation of high-level synthesis. *IEEE Trans on CAD of ICS*, 29(4):566–579, 2010.

[4] G. C. Necula. Translation validation for an optimizing compiler. In *PLDI*, pages 83–94, 2000.