Algorithms for Model Checking

Having demonstrated the benefits of formal verification, we proceed to describe algorithms and methods by which such verification can be accomplished.

A run of $\mathcal{D}$ is a finite or infinite state sequence which satisfies the requirements of initiality and consecution but is not necessarily fair.

A run segment is a finite state sequence which satisfies the requirement of consecution.

A state s is $\mathcal{D}$-accessible if it appears in some $\mathcal{D}$-run.

System $\mathcal{D}$ is finite-state if it has only finitely many accessible states. An SPL program with a fixed number of processes such that all of its variables are declared to range over a finite domain (boolean or enumerated type) corresponds to a finite-state FDS.

We start by presenting algorithms for the verification over finite-state systems of the following two classes of properties:

- The invariance property $Inv(p)$, claiming that all $\mathcal{D}$-accessible states satisfy the assertion p.
- The response property $p \rightsquigarrow q$, claiming that every ($\mathcal{D}$-accessible) p-state must be followed by a q-state.

The State-Transition Graph

A state-transition graph (S, E) is a directed graph whose nodes S are states of some system $\mathcal{D}$ and whose edges E connect state s to state $\widetilde{s}$ iff $\widetilde{s}$ is a $\rho_{\mathcal{D}}$-successor of s.

The following algorithm constructs the state-transition graph $G(S_0, \rho)$ which contains all the states reachable from the set S_0 by ρ-transitions.

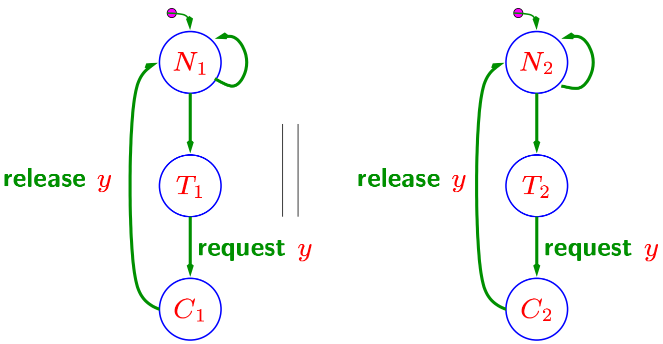
Algorithm CONSTRUCT-GRAPH(S_0, ρ) — construct the state-transition graph $G(S_0, \rho)$

- Initially place in S all states that are in S_0.
- Repeat the following step until no new states or new edges can be added to G.
  - **Step:** for some $s \in S$, let $s_1, \ldots, s_k$ be the ρ-successors of s. Add to S all states among $\{s_1, \ldots, s_k\}$ which are not already there and add to E edges connecting s to $s_1, \ldots, s_k$.
- Return (S, E)

Example: a Simpler MUX-SEM

Below, we present a simpler version of program MUX-SEM.

y: **natural initially** $y = 1$

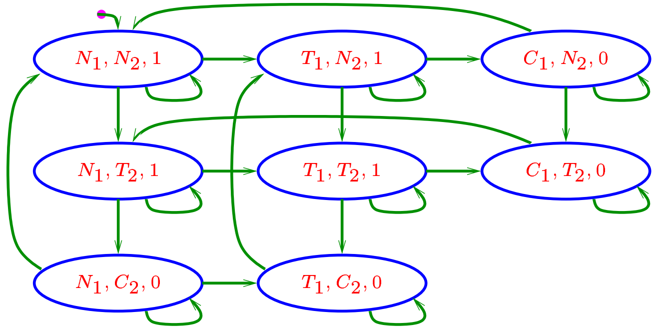


The semaphore instructions **request** y and **release** y respectively stand for

$$\langle \textbf{when } y = 1 \textbf{ do } y := 0 \rangle \quad \text{and} \quad y := 1.$$

The state-transition Graph for MUX-SEM

Following is the state-transition graph $G(\|\Theta\|, \rho)$ for MUX-SEM. This graph contains all the states accessible by MUX-SEM. Here and elsewhere, we denote by $\|p\|$ the set of states satisfying p. Thus, $\|\Theta\| = \{(N_1, N_2, 1)\}$ is the set of initial states of MUX-SEM.



Model Checking Invariance Properties

We may use the following algorithm to verify that system $\mathcal{D}$ satisfies the invariance property $Inv(p)$.

Algorithm MC-INV$(\mathcal{D}, p)$ — verify that p is an invariant of system $\mathcal{D}$

- Let $(S, E) :=$ CONSTRUCT-GRAPH$(\|\Theta\|, \rho)$
- Search in S for a state s violating the assertion p.
- If no such state found, print "Property is Valid".
- Otherwise, print the (shortest) path leading from some Θ-state to the violating state s, indicating "Property is Invalid".

Using this algorithm, we can ascertain that program MUX-SEM satisfies the invariance property of mutual exclusion, given by $Inv(\neg(C_1 \wedge C_2))$.

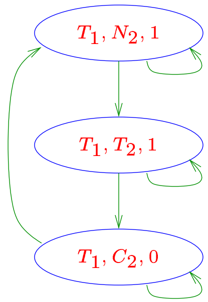
Now to Response Properties

Next, we consider an algorithm for verifying response properties. A state s is defined to be pending if it is reachable by a q-free path from a state $\widetilde{s}$ which is an accessible p-state.

We start by forming the state-transition graph G_{pend} which consists of all the pending states. This can be done by the following operations:

$$\begin{array}{lll} \rho_{\neg q} & := & \rho \wedge \neg q \wedge \neg q' \\ (S, E) & := & \text{CONSTRUCT-GRAPH}(\|\Theta\|,\ \rho) \\ (S_{pend}, E_{pend}) & := & \text{CONSTRUCT-GRAPH}(S \cap \|p \wedge \neg q\|,\ \rho_{\neg q}) \end{array}$$

For example, considering program MUX-SEM under the response property $T_1 \rightsquigarrow C_1$, we obtain the following graph as capturing all the pending states:



A fair path in a state-transition graph is an infinite path which satisfies the two classes of fairness requirements.

Observation 1. *System $\mathcal{D}$ violates the response property $p \rightsquigarrow q$ iff the graph G_{pend} contains a fair path.*

Thus, it is sufficient to check whether G_{pend} contains a fair path.

From Fair Paths to Fair Subgraphs

A subgraph $S \subseteq G_{pend}$ is called a strongly connected subgraph (SCS) if, for every two distinct states $s_1, s_2 \in S$, there exists a path from s_1 to s_2 which only traverses states of S. For example, $\{\langle N_1, N_2, 1\rangle, \langle T_1, N_2, 1\rangle, \langle C_1, N_2, 0\rangle\}$, and $\{\langle N_1, N_2, 1\rangle\}$ are both SCS's of the state-transition graph of MUX-SEM. An SCS is called singular if it consists of a single state which is not connected to itself.

A subgraph S is called just if it contains a J-state for every justice requirement $J \in \mathcal{J}$. The subgraph S is called compassionate if, for every compassion requirement $(p, q) \in \mathcal{C}$, S contains a q-state, or S contains no p-state.

A subgraph S is fair if it is a non-singular strongly connected subgraph which is both just and compassionate.

Let π be an infinite path in G_{pend}. We denote by $\mathit{Inf}(\pi)$ the set of states which appear infinitely many times in π.

Traversing Cycles within SCSs

Observation 2. *Every strongly connected subgraph S contains a traversing cyclic path $\pi : s_0, s_1, \ldots, s_k = s_0$ which visits each state of S at least once.*

Proved by construction. Start by $\pi : s_0$, where $s_0 \in S$ is an arbitrary state in S. Let $\mathit{last}(\pi)$ denote the last state in the path π.

While $S - set(\pi) \neq \emptyset$ **do**

- Choose $s \in S - set(\pi)$.
- Let κ be an S-path connecting $last(\pi)$ to s. Guaranteed to exists due to the strong connectedness of S.
- Append κ to the end of π

Finally, extend π by a path connecting $last(\pi)$ to s_0.

A necessary and Sufficient Condition

The following claim connects fair paths within G_{pend} with fair subgraphs of G_{pend}.

Claim 2. *The graph G_{pend} contains a fair path iff it contains a a fair subgraph.*

Fair path $\Longrightarrow$ fair subgraph
Let π be a fair path within G_{pend}. We will show that $S = \mathit{Inf}(\pi)$ is a fair subgraph.

Note that there exists a position $j \geq 0$ such that every state that appears in a pi beyond position j belongs to $\mathit{Inf}(\pi)$ and, therefore appears infinitely many time beyond j.

Let $s^a, s^b \in S$. Since both states appear infinitely many times beyond j, there exists positions $j < k < m$, such that $s_k = s^a$ and $s_m = s^b$. The sequence $s_k, s_{k+1}, \ldots, s_{m-1}, s_m$ is a path within G_{pend} which only visits states occurring at positions beyond j. Therefore, it is a path within S leading from s^a to s^b. This shows that $S = \mathit{Inf}(\pi)$ is a non-singular strongly connected subgraph of G_{pend}.

Let J_i be one of the justice requirements. Since π is fair, it contains infinitely many J_i-states. In particular (since G_{pend} is finite) there must exists a particular J_i-state s^i which appears infinitely many times in π. Obviously $s^i \in \mathit{Inf}(\pi) = S$.

Let (p_i, q_i) be one of the compassion requirements. Since π is fair, it either contains only finitely many p_i-states or contains infinitely many q_i-states. In the first case, $S = \mathit{Inf}(\pi)$ contains no p_i-states. In the second case, $S = \mathit{Inf}(\pi)$ contains at least one q_i-state.

Course G22.3033.007 Lecture 3 R. Dewar and A. Pnueli

Fair Subgraph $\Longrightarrow$ Fair Path

Assume that $S \subseteq G_{pend}$ is a fair subgraph. Let κ be the cycle traversing all states of S. We denote by $\pi = \kappa^\omega$ the infinite path obtained by infinite repetition of the cycle κ.

We claim that π is a fair path. For every justice requirement J_i, S contains some J_i-state s^i. Since κ passes through s^i at least once, $\pi = \kappa^\omega$ visits s^i infinitely many times.

Similarly, let (p_i, q_i) be a compassion requirement. Either S contains no p_i-states at all, in which case, neither does π. Alternately, S contains some q_i-state s^i, in which case, $\pi = \kappa^\omega$ contains infinitely many copies of s^i. ▟

Corollary 3. *A system $\mathcal{D}$ violates the response property $p \leadsto q$ iff G_{pend} contains a fair subgraph.*

A subgraph S is called a maximal strongly connected subgraph (MSCS), if S is strongly connected and is not properly contained in any larger SCS.

There exists an algorithm (due to Tarjan) , which decomposes a given graph into a list of MSCSs,

$$G_{pend} = S_1 \cup S_2, \cup \cdots \cup S_k,$$

such that an edge can connect a state in S_i with a state in S_j only if $i \leq j$.

In Search of Fair Subgraphs

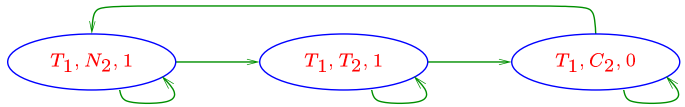
The following recursive algorithm accepts as input an SCS S and returns a fair subgraph of S if one exists, or the empty set if S contains no fair subgraph. Here and elsewhere, we denote by $\|p\|$ the set of all p-states.

Algorithm FAIR-SUB(S : **set**) : **set** — Find a fair subgraph within S

- **if** S is singular **then return** $\emptyset$ -- failure
- **if** S is not just **then return** $\emptyset$ -- failure
- **if** S is compassionate **then return** S -- success
- -- S is just but not compassionate. Let $\widetilde{C} \subseteq \mathcal{C}$ be
  -- the set of all compassion requirement (p_i, q_i) such
  -- that S contains no q_i-states.
- **let** $U = S - \bigcup_{(p_i,q_i)\in\widetilde{C}} \|p_i\|$.
- Decompose U into MSCS's $U_1, \ldots, U_k$.
- **let** $V = \emptyset$, $i = 1$
- **while** $V = \emptyset$ and $i \leq k$ **do**
  - **let** V = FAIR-SUB(U_i)
  - $i := i + 1$
- **return** V

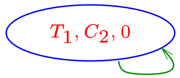
Example

Reconsider the pending graph G_{pend} for the response property $T_1 \leadsto C_1$ over program MUX-SEM.



Applying algorithm FAIR-SUB to this graph, we find that G_{pend} is non-singular and just. However, it is not compassionate w.r.t requirement $(T_1 \wedge y > 0, C_1)$.

We therefore remove from the graph all states which satisfy $T_1 \wedge y > 0$. This leaves us with



which is non-singular but unjust towards the justice requirement $\neg C_2$. We conclude that G_{pend} contains no fair subgraphs and, therefore, the property $T_1 \leadsto C_1$ is valid over MUX-SEM.

Model Checking Response Properties

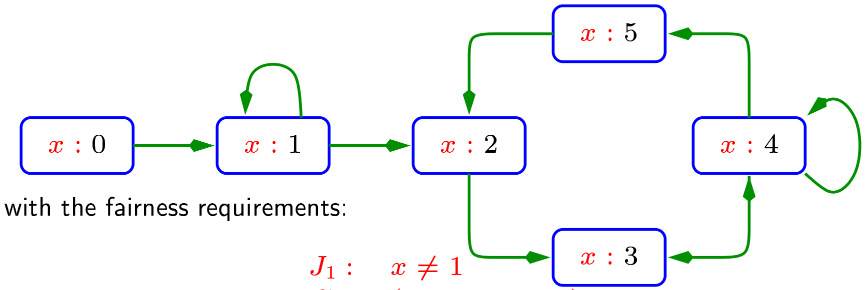
Finally, we present the algorithm that checks whether a given FDS $\mathcal{D}$ satisfies a response property $p \rightsquigarrow q$. This is achieved by the following algorithm which accepts as input an FDS $\mathcal{D}$ and two assertions p and q, returning an empty set (graph) iff $\mathcal{D}$ satisfies $p \rightsquigarrow q$.

Algorithm MC-RESP($\mathcal{D}$: FDS; p, q : assertion) : **set** — Check whether FDS $\mathcal{D}$ satisfies $p \rightsquigarrow q$

- Invoke algorithm CONSTRUCT-GRAPH to compute G_{pend} the pending graph for system $\mathcal{D}$.
- Decompose G_{pend} into MSCS's $S_1, \ldots, S_k$.
- **let** $V = \emptyset, \quad i = 1$
- **while** $V = \emptyset$ and $i \leq k$ **do**
  - **let** V = FAIR-SUB(S_i)
  - $i := i + 1$
- **return** V

Example

As an example, consider the following FDS:



with the fairness requirements:

$$J_1 : \quad x \neq 1$$
$$C_1 : \quad (x = 3,\ x = 5)$$
$$C_2 : \quad (x = 2,\ x = 1)$$

The initial decomposition into MSCS's yields the partition

$\{s_0\}, \quad \{s_1\}, \quad \{s_2, s_3, s_4, s_5\}$.

Applying FAIR-SUB to these subgraphs, we get

FAIR-SUB($\{s_0\}$) $= \emptyset$ because $\{s_0\}$ is singular
FAIR-SUB($\{s_1\}$) $= \emptyset$ because $\{s_1\}$ is unjust

Applied to $\{s_2, s_3, s_4, s_5\}$, FAIR-SUB finds that $\{s_2, s_3, s_4, s_5\}$ is non-singular, just, and compassionate w.r.t C_1. However, it is in-compassionate w.r.t C_2.

Therefore, we remove s_2 and proceed to apply FAIR-SUB to the decomposition of $\{s_3, s_4, s_5\}$, which is $\{\{s_3, s_4\}, \quad \{s_5\}\}$.

SCS $\{s_3, s_4\}$ is in-compassionate towards C_1 which causes us to remove s_3. We are left with $\{s_4\}$ which is non-singular, just and compassionate towards both C_1 and C_2. Therefore, the algorithm returns $\{s_4\}$ as a fair subgraph of the system.