Recognizing Safety and Liveness

Stefano Pessotto

7 November 2024

University of Udine

Table of contents

Model Checking

In the context of model checking we want to show that the system behaves correctly in every possible scenario.

$$\pi \models P$$

In the general case, the problem is hard and the complexity depends on the logic we choose to express the properties.

Programs

$$\pi \models P$$

Program

π is a program, formally represented in terms of:

- A set of program states S_π;
- A set of atomic actions A_π;
- A predicate for the initial states $Init_\pi$.

Programs

- S_π: The only requirement for S_π is to be a countable set.
- A_π: Atomic actions are subsets of $S_\pi \times S_\pi$.

$$\begin{aligned}\alpha &= < \mathit{if}\ b \rightarrow C\ \mathit{fi} > \\ &= \{(s,t) \in S_\pi \times S_\pi : s \models b \wedge t = C(s)\}\end{aligned}$$

  α is enabled in s if there exists a state t such that $(s,t) \in \alpha$.
- $Init_\pi$: Any state s such that $s \models Init_\pi$ is a possible initial state.

History

A sequence of states $\sigma = s_0 s_1 \ldots$ is called a history of π whenever

- $s_0 \models Init_\pi$;
- Every state s_{i+1} is the result of the execution of a single enabled atomic action from A_π in s_i.

Finite executions are extended into infinite ones by repeating the last occurring state.

Properties

$$\pi \models P$$

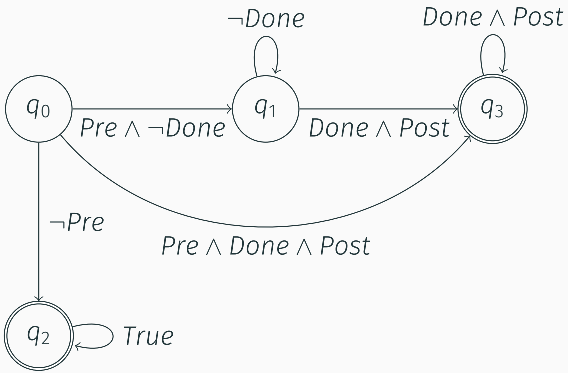
$\mathcal{M}_P$

Properties can be expressed in terms of Büchi Automaton $\mathcal{M}_P$:

$$\pi \models P \iff \forall \sigma \text{ history of } \pi, \sigma \in \mathcal{L}(\mathcal{M}_P)$$

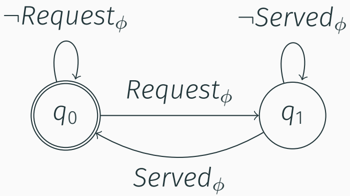
The expressiveness of Büchi Automaton is equivalent to ETL.

Properties: Total Correctness



- *Pre* holds for states satisfying the preconditions;
- *Done* holds for states in which the program has terminated;
- *Post* holds for states satisfying the post-conditions.

Properties: Starvation Freedom



- $Request_\phi$ holds for states in which process ϕ require access to the critical sections;
- $Served_\phi$ holds for states in which process ϕ enters the critical section.

Property automaton

$\mathcal{M}_{(P,\pi)}$

A Büchi Automaton $\mathcal{M}_{(P,\pi)}$ is a quintuple $< S_\pi, Q, Q_0, Q_\infty, \delta >$ where

- S_π is the set of states of the program π;
- Q is the set of automaton states of the property $\mathcal{M}_P$;
- $Q_0 \subseteq Q$ is the set of initial states of $\mathcal{M}_P$;
- $Q_\infty \subseteq Q$ is the set of accepting states of $\mathcal{M}_P$;
- $\delta : (Q \times S_\pi) \to 2^Q$ is the transition function.

Property automaton

Given an execution σ of π:

- $\Gamma_{\mathcal{M}_{(P,\pi)}}(\sigma)$: set of all runs of $\mathcal{M}_{(P,\pi)}$ over σ;
- $INF_{\mathcal{M}_{(P,\pi)}}(\sigma)$: set of automaton states that appears infinitely many times in any element of $\Gamma_{\mathcal{M}_{(P,\pi)}}(\sigma)$;
- σ is accepted if and only if $INF_{\mathcal{M}_{(P,\pi)}} \cap Q_\infty \neq \varnothing$.

Reduced Büchi Automaton

A Büchi Automaton is reduced if and only if there is a path from every state to an accepting state.

Any Büchi Automaton $\mathcal{M}$ is equivalent to a reduced one, and it can be obtained by simply removing all states from which non accepting state is reachable.

Closure Automaton

The closure automaton $cl(\mathcal{M})$ of the Büchi Automaton $\mathcal{M}$ is the automaton in which every state becomes a final state.

The closure automaton can rejects ω-words only by attempting undefined transitions.

Safety

Safety properties express the fact that ”*something bad never happens*”, which means that any violation is irremediable and has a finite witness.
In the context of automaton, the violation is the attempt to take an undefined transition:

- It is irremediable: halts the computation;
- It has a finite witness: the prefix up to the time point in which the undefined transition is taken defines the witness.

Safety

Safety

A property P is a safety property if and only if

$$\forall \sigma \in S_\pi^\omega (\sigma \models P \iff (\forall i \geq 0\ \exists \beta \in S_\pi^\omega : \sigma[\ldots i] \cdot \beta \models P))$$

The definition is the contrapositive of

$$\forall \sigma \in S_\pi^\omega (\sigma \not\models P \iff (\exists i \geq 0\ \forall \beta \in S_\pi^\omega : \sigma[\ldots i] \cdot \beta \not\models P))$$

Safety

The closure automaton can be used to check whether P is safety, since it can reject an input only by taking an undefined transition: whenever $\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(cl(\mathcal{M}_P))$ holds, we have that $\mathcal{M}_P$ rejects input only by taking undefined transition.

Theorem

A reduced Büchi Automaton $\mathcal{M}_P$ specifies a safety property if and only if

$$\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(cl(\mathcal{M}_P))$$

Safety

Proof: $\Rightarrow$
It holds by definition that $\mathcal{L}(\mathcal{M}_P) \subseteq \mathcal{L}(cl(\mathcal{M}_P))$.
To show that $\mathcal{L}(cl(\mathcal{M}_P)) \subseteq \mathcal{L}(\mathcal{M}_P)$ we apply the definition of safety:
Let $\alpha \in \mathcal{L}(cl(\mathcal{M}_P))$ and $i \in \mathbb{N}$

$$\exists \beta \in S^\omega : \alpha[\ldots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P)$$

Let $q_i = \delta^*(q_0, \alpha[\ldots i])$

- $\mathcal{M}_P$ is reduced, so q_i precedes an accepting state;
- $\alpha \in \mathcal{L}(cl(\mathcal{M}_P))$, so both $cl(\mathcal{M}_P)$ and $\mathcal{M}_P$ do not take an undefined transition;

so there exists an extension β_0 such that $\delta^*(q_i, \beta_0) = q_f$ and $q_f \in Q_\infty$.

Safety

Proof: $\Rightarrow$

$$\dots$$

This argument can be iterated to build an infinite suffix $\beta = \beta_0 \cdot \beta_1 \cdot \dots$ such that $\alpha[\dots i] \cdot \beta$ visits at least one accepting state infinitely often, so $\alpha[\dots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P)$.
From the facts that

- $\mathcal{M}_P$ specifies a safety property $\implies \forall \alpha \in \mathcal{L}(cl(\mathcal{M}_P))$

$$(\forall i \geq 0 \; \exists \beta \in S^\omega : \alpha[\dots i] \cdot \beta \models P) \Rightarrow \alpha \models P$$

- $\mathcal{M}_P$ specifies a safety property
- $\forall i \geq 0 \; \exists \beta \in S^\omega : \alpha[\dots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P)$

it follows that $\alpha \models P$, or equivalently $\alpha \in \mathcal{L}(\mathcal{M}_P)$. $\square$

Safety

Proof: $\Leftarrow$

Assume $\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(cl(\mathcal{M}_P))$. In order to prove that $\mathcal{M}_P$ specifies a safety property, we will prove

$$\sigma \in \mathcal{L}(\mathcal{M}_P) \iff \forall i \geq 0\, \exists \beta \in S^\omega \sigma[\ldots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P)$$

($\Longrightarrow$) is trivially satisfied we can choose $\beta = \sigma[i+1\ldots]$.

We want to prove

$$\forall i \geq 0\, \exists \beta \in S^\omega \sigma[\ldots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P) \implies \sigma \in \mathcal{L}(\mathcal{M}_P)$$

equivalent to

$$\sigma \notin \mathcal{L}(\mathcal{M}_P) \implies \neg(\forall i \geq 0\, \exists \beta \in S^\omega (\sigma[\ldots i]\beta \in \mathcal{L}(\mathcal{M}_P))$$

$$\sigma \notin \mathcal{L}(\mathcal{M}_P) \implies (\exists i \geq 0\, \forall \beta \in S^\omega (\sigma[\ldots i] \cdot \beta \notin \mathcal{L}(\mathcal{M}_P))$$

Safety

$$\ldots$$

$$\sigma \notin \mathcal{L}(\mathcal{M}_P) \implies (\exists i \geq 0\, \forall \beta \in S^\omega (\sigma[\ldots i] \cdot \beta \notin \mathcal{L}(\mathcal{M}_P))$$

Since $\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(cl(\mathcal{M}_P))$, we can substitute the two on both sides

$$\sigma \notin \mathcal{L}(cl(\mathcal{M}_P)) \implies (\exists i \geq 0\, \forall \beta \in S^\omega (\sigma[\ldots i] \cdot \beta \notin \mathcal{L}(cl(\mathcal{M}_P)))$$

Let $\alpha \notin \mathcal{L}(cl(\mathcal{M}_P))$, then α is rejected upon taking an undefined transition because all states in $cl(\mathcal{M}_P)$ are finals. Let k be the index of such transition. It is easy to see that $\forall \beta \in S^\omega$

$$\sigma[\ldots k] \cdot \beta \notin \mathcal{L}(cl(\mathcal{M}_P))$$

$\square$

Liveness

Liveness properties express the fact that *”something good will eventually happen”*.
In the context of automaton, this can be recognized as the ability to recover any partial execution and is the opposite of Safety.
Liveness properties do not allow *”bad things”* to happen, since they would be irrecoverable: any violation requires an infinite behavior and is not detectable by analyzing a finite prefix.

Liveness

A reduced Büchi Automaton $\mathcal{M}_P$ specifies a liveness property if and only if

$$\forall \alpha \in S^* \, \exists \beta \in S^\omega (\alpha \cdot \beta \in \mathcal{L}(\mathcal{M}_P))$$

Notice that the definition is different from *CoSafety*, which states that

$$\forall \sigma \in \mathcal{L}(\mathcal{M}_P) \, \exists i \in \mathbb{N} \, \forall \beta \in S^\omega (\sigma[\ldots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P))$$

and requires only finite time to check the satisfaction of an ω-word.

Liveness

The closure automaton can be used to check whether a property is liveness, since it can reject an input only by taking an undefined transition. For P to be liveness, $\mathcal{M}_P$ must not take any undefined transition, which means that the language recognized by the closure automaton must be S^ω.

Theorem

A reduced Büchi Automaton $\mathcal{M}_P$ specifies a liveness property if and only if

$$\mathcal{L}(cl(\mathcal{M}_P)) = S^\omega$$

Liveness

Proof: $\Rightarrow$

Assume $\mathcal{M}_P$ specifies a liveness property and $\alpha \in S^\omega$. We can instantiate the definition of liveness for any prefix of α:

$$\forall i \geq 0 \; \exists \beta \in S^\omega (\alpha[\dots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P))$$

meaning that $\mathcal{M}_P$ does not take any undefined transition reading α, otherwise it would be irrecoverable.

Since $\mathcal{M}_P$ and its closure share the same transition function, none of them take an undefined transition and the closure automaton must accept α.

This means that $\forall \alpha \in S^\omega \; \alpha \in \mathcal{L}(cl(\mathcal{M}_P)) = S^\omega$. $\square$

Liveness

Proof: $\Leftarrow$

Assume $\mathcal{L}(cl(\mathcal{M}_P)) = S^\omega$ and $\alpha \in S^\omega$. We want to prove that $\mathcal{M}_P$ specifies a liveness property by showing that $\forall i \in \mathbb{N}$

$$\exists \beta \in S^\omega : \alpha[\ldots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P))$$

Let $i \in \mathbb{N}$ and $q_i = \delta^*(q_0, \alpha[\ldots i])$

- $\mathcal{M}_P$ is reduced, so q_i precedes an accepting state;
- Since $\mathcal{L}(cl(\mathcal{M}_P)) = S^\omega$, $cl(\mathcal{M}_P)$ does not take any undefined transition while reading $\alpha[\ldots i]$, and the same behavior applies to $\mathcal{M}_P$;

so there exists an extension β_0 such that $\delta^*(q_i, \beta_0) = q_f$ and $q_f \in Q_\infty$.

Liveness

Proof: $\Leftarrow$

$$\dots$$

This argument can be iterated to build an infinite suffix $\beta = \beta_0 \cdot \beta_1 \cdot \dots$ such that $\alpha[\dots i] \cdot \beta$ visits at least one accepting state infinitely often, so $\alpha[\dots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P)$.
It follows that

$$\exists \beta \in S^\omega : \alpha[\dots i] \cdot \beta \in \mathcal{L}(\mathcal{M}_P)$$

so the definition of liveness holds. $\square$

Partitioning Properties

We now want to show that any property P, expressed by an automaton $\mathcal{M}_P$, can be partitioned into two automata such that

- A Büchi Automaton $\mathit{Safe}(\mathcal{M}_P)$ will specify the safety part of P;
- A Büchi Automaton $\mathit{Live}(\mathcal{M}_P)$ will specify the liveness part of P;
- P will be described by the intersection of $\mathit{Safe}(\mathcal{M}_P)$ and $\mathit{Live}(\mathcal{M}_P)$.

$Safe(\mathcal{M}_P)$

The closure automaton can be exploited.

Theorem

$Safe(\mathcal{M}_P) = cl(\mathcal{M}_P)$ specifies a safety property.

The proof is trivial since $\mathcal{L}(cl(\mathcal{M}_P)) = \mathcal{L}(cl(cl(\mathcal{M}_P)))$.

Theorem: recall

A reduced Büchi Automaton $\mathcal{M}_P$ specifies a safety property if and only if

$$\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(cl(\mathcal{M}_P))$$

$Live(\mathcal{M}_P)$: deterministic

The objective is to construct an automaton such that

$$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$$

We need to distinguish the deterministic and non deterministic case. In the deterministic case

- A new accepting trap state q_{trap} is added to $\mathcal{M}_P$;
- every undefined transition is replaced with a transition into q_{trap}.

$Live(\mathcal{M}_P)$: deterministic

Lemma

$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$

Proof: $\subseteq$

Assume $\alpha \in \mathcal{L}(Live(\mathcal{M}_P))$, one of the two cases occurs

- $q_{trap} \in INF_{\mathcal{M}_P}(\alpha)$, which means that $\mathcal{M}_P$ would have taken an undefined transition while reading α. Since the closure automaton behaves like $\mathcal{M}_P$,
  $\alpha \notin \mathcal{L}(cl(\mathcal{M}_P)) \equiv \alpha \in S^\omega - \mathcal{L}(cl(\mathcal{M}_P))$;

Lemma

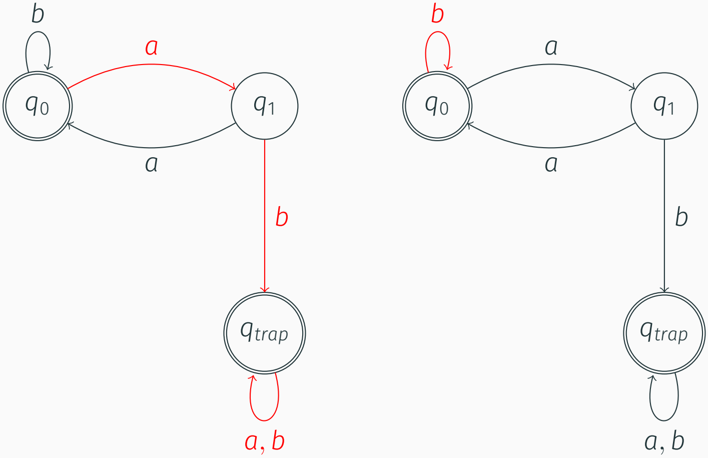
$$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$$

Proof: $\subseteq$

Assume $\alpha \in \mathcal{L}(Live(\mathcal{M}_P))$, one of the two cases occurs

- $q_{trap} \notin INF_{\mathcal{M}_P}(\alpha)$, which means that another state $q_f \in Q_\infty$ occurs infinitely many times in the computation of $Live(\mathcal{M}_P)$, and so it does in the computation of $\mathcal{M}_P$, which means that $\alpha \in \mathcal{L}(\mathcal{M}_P)$.

$Live(\mathcal{M}_P)$: deterministic



$Live(\mathcal{M}_P)$: deterministic

Lemma

$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$

Proof: $\supseteq$
Assume $\alpha \in \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$

- if $\alpha \in \mathcal{L}(\mathcal{M}_P)$, then $\alpha \in \mathcal{L}(Live(\mathcal{M}_P))$ by construction, since they are equivalent when no undefined transition is taken;

$Live(\mathcal{M}_P)$: deterministic

Lemma

$$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$$

Proof: $\supseteq$

Assume $\alpha \in \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$

- if $\alpha \in S^\omega - \mathcal{L}(cl(\mathcal{M}_P))$, then $\alpha \notin \mathcal{L}(cl(\mathcal{M}_P))$
  - $cl(\mathcal{M}_P)$ takes an undefined transition while reading α, and so does $\mathcal{M}_P$;
  - That transition is replaced by a transition into q_{trap} in $Live(\mathcal{M}_P)$;
  - q_{trap} is an accepting trap state, and will be visited infinitely many times.

$\square$

$Live(\mathcal{M}_P)$: non deterministic

The objective is to construct an automaton such that

$$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$$

We need to distinguish the deterministic and non deterministic case. In the non deterministic case

- The automaton $env(\mathcal{M}_P)$ for $S^\omega - \mathcal{L}(cl(\mathcal{M}_P))$ is constructed
  - Apply the subset construction to determinize $\mathcal{M}_P$;
  - Add a new trap state q_{trap}, which will be the only accepting state;
  - Replace every undefined transition with a transition into q_{trap}.
- $Live(\mathcal{M}_P)$ is obtained by the union of $\mathcal{M}_P$ and $env(\mathcal{M}_P)$.

$Live(\mathcal{M}_P)$: non deterministic

Lemma

$$\mathcal{L}(Live(\mathcal{M}_P)) = \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))$$

Since $env(\mathcal{M}_P)$ specifies $S^\omega - \mathcal{L}(cl(\mathcal{M}_P))$ by capturing all undefined transitions in $cl(\mathcal{M}_P)$, the proof is trivial by closure properties for non deterministic Büchi Automaton.

Theorem

$Live(\mathcal{M}_P)$ specifies a liveness property.

The proof is trivial in the deterministic case, since all undefined transition are replaced by a transition into q_{trap} by construction.

Theorem: recall

A reduced Büchi Automaton $\mathcal{M}_P$ specifies a liveness property if and only if

$$\mathcal{L}(cl(\mathcal{M}_P)) = S^\omega$$

Theorem

$Live(\mathcal{M}_P)$ specifies a liveness property.

In the non deterministic case, from the previous lemma we have that

$$
\begin{aligned}
\mathcal{L}(Live(\mathcal{M}_P)) &= \mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P))) \\
\mathcal{L}(cl(Live(\mathcal{M}_P))) &= \mathcal{L}(cl(\mathcal{M}_P)) \cup \mathcal{L}(cl(env(\mathcal{M}_P))) \\
\mathcal{L}(cl(Live(\mathcal{M}_P))) &\supseteq \mathcal{L}(cl(\mathcal{M}_P)) \cup \mathcal{L}(env(\mathcal{M}_P)) \\
\mathcal{L}(cl(Live(\mathcal{M}_P))) &\supseteq \mathcal{L}(cl(\mathcal{M}_P)) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P))) \\
\mathcal{L}(cl(Live(\mathcal{M}_P))) &\supseteq S^\omega \\
\mathcal{L}(cl(Live(\mathcal{M}_P))) &= S^\omega
\end{aligned}
$$

$\square$

Theorem

Given a reduced Büchi Automaton $\mathcal{M}_P$

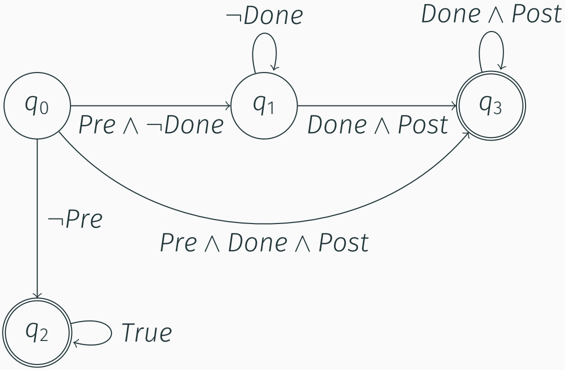
$$\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(Safe(\mathcal{M}_P)) \cap \mathcal{L}(Live(\mathcal{M}_P))$$

Proof:

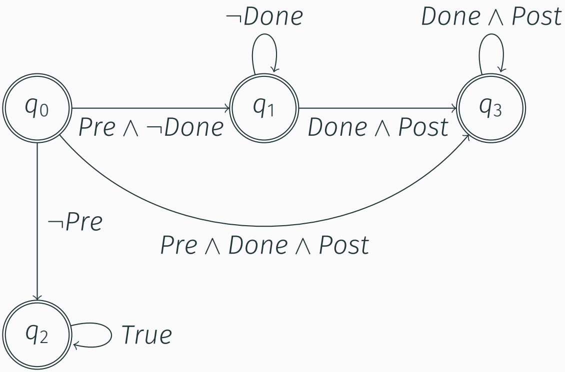
$$\begin{aligned}
&\mathcal{L}(Live(\mathcal{M}_P)) \cap \mathcal{L}(Safe(\mathcal{M}_P))\\
&(\mathcal{L}(\mathcal{M}_P) \cup (S^\omega - \mathcal{L}(cl(\mathcal{M}_P)))) \cap \mathcal{L}(cl(\mathcal{M}_P))\\
&(\mathcal{L}(\mathcal{M}_P) \cap \mathcal{L}(cl(\mathcal{M}_P))) \cup ((S^\omega - \mathcal{L}(cl(\mathcal{M}_P))) \cap \mathcal{L}(cl(\mathcal{M}_P)))\\
&(\mathcal{L}(\mathcal{M}_P) \cap \mathcal{L}(cl(\mathcal{M}_P))) \cup \varnothing)\\
&\mathcal{L}(\mathcal{M}_P)
\end{aligned}$$

$\square$

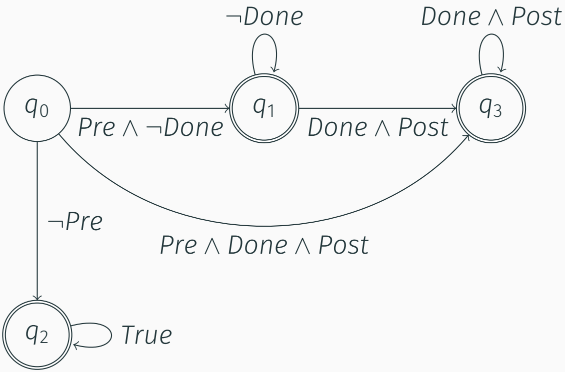
Example: $\mathcal{M}_{\mathit{Total\ Correctness}}$



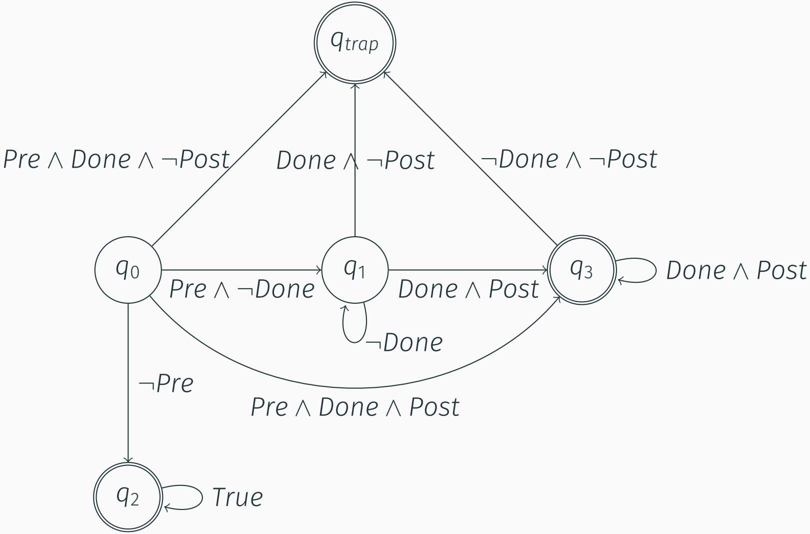
Example: $Safe(\mathcal{M}_{Total\ Correctness})$



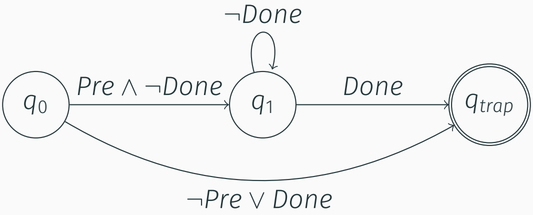
Example: $\mathcal{L}(Safe(\mathcal{M}_{Total\ Correctness})) = \mathcal{L}(\mathcal{M}_{Partial\ Correctness})$



Example: $Live(\mathcal{M}_{Total\ Correctness})$



Example: $\mathcal{L}(\mathit{Live}(\mathcal{M}_{\mathit{Total\ Correctness}})) = \mathcal{L}(\mathcal{M}_{\mathit{Termination}})$



Program Verification for deterministic properties

To verify whether $\pi \models P$, it needs to be proven that every possible history of π is accepted by the automaton $\mathcal{M}_{(P,\pi)}$.
In order to do so, we will specify proof obligations using Hoare's logic, which specifies triples of the form $\{Pre\}action\{Post\}$, where

- *Pre* is the set of preconditions;
- *action* is the code fragment that must terminate;
- *Post* is the set of postconditions.

Correspondence invariant

Correspondence invariant

Let q_i be an automaton state and s be a program state.
A correspondence invariant C_i for q_i is a predicate such that $s \models C_i$ if and only if there exists a history of π containing s, and $\mathcal{M}_{(P,\pi)}$ enters q_i upon reading s.

Correspondence invariants are needed to maintain the consistency between program states and automaton states. A correspondence invariant C_i is defined for each automaton state q_i.

Reject knot and Variant function

Reject knot

A *reject knot* $\mathcal{K}$ is a maximal strongly connected subset of automaton states in $\mathcal{M}_{(P,\pi)}$ with no accepting states.

Variant function

Let q_i be an automaton state and s be a program state. Let $\mathcal{K}$ be a reject knot.
A variant function $v_{\mathcal{K}}(q_i, s)$ is a function from $Q \times S_\pi$ to a well-founded set, such as $\mathbb{N}$.

The variant function is used to keep track of the evolution of the computation.

Transition predicate

Transition predicate

A *transition predicate* T_{ij} for $\mathcal{M}_{(P,\pi)}$ is a predicate that holds for all program states $s \in S_\pi$ such that $q_j \in \delta(q_i, s)$.

Transition predicates are used in proof obligations to track the possible computations of the automaton.

Proof obligations

Correspondence Basis

$$\forall j : q_j \in Q(Init_\pi \wedge T_{0j} \implies C_j)$$

Correspondence Induction

$$\forall \alpha \in A_\pi \, \forall i : q_i \in Q(\{C_i\}\alpha\{\bigwedge_{j:q_j \in Q}(T_{ij} \implies C_j)\})$$

They impose the correctness of the correspondence invariants.

Proof obligations

Transition Basis

$$Init_\pi \implies \bigvee_{j:q_j \in Q} T_{0j}$$

Transition Induction

$$\forall \alpha \in A_\pi \; \forall i : q_i \in Q(\{C_i\}\alpha\{\bigvee_{j:q_j \in Q} T_{ij}\})$$

Enforce the safety part of $\mathcal{M}_P$ as they force the automaton to avoid undefined transition.

Proof obligations

Knot Exit

$$\forall \mathcal{K}\, \forall i : q_i \in \mathcal{K} (v_{\mathcal{K}}(q_i) = 0 \implies \neg C_i)$$

Knot Variance

$$\forall \mathcal{K}\, \forall \alpha \in A_\pi\, \forall q_i \in \mathcal{K}$$

$$(\{C_i \wedge (0 < v_{\mathcal{K}}(q_i) = V)\}\alpha\{\bigwedge_{j:q_j \in \mathcal{K}} ((T_{ij} \wedge C_j) \implies (v_{\mathcal{K}}(q_j) < V))\})$$

Enforce the liveness part of $\mathcal{M}_P$ as they impose the termination of π in possible infinite loops with no accepting states.

Obligations for Safety

When dealing with safety properties:

- *Safe*($\mathcal{M}_P$) doesn't have any reject knots: obligations *Knot Exit* and *Knot Variance* are trivially satisfied;
- For every safety property $\mathcal{L}(\mathcal{M}_P) = \mathcal{L}(\mathit{Safe}(\mathcal{M}_P))$: proving $\pi \models \mathit{Safe}(\mathcal{M}_P)$ is sufficient to prove that $\pi \models \mathcal{M}_P$.

Proof obligations *Correspondence Basis*, *Correspondence Induction*, *Transition Basis* and *Transition Induction* needs to be checked, and they involve an *invariance argument*.

Obligations for Liveness

When dealing with liveness properties:

- For any liveness property, the corresponding automaton cannot take any undefined transition, since $\mathcal{L}(cl(\mathcal{M}_P)) = S^\omega$: obligations *Transition Basis* and *Transition induction* are trivially satisfied.

Proof obligations *Correspondence Basis*, *Correspondence Induction*, *Knot Exit* and *Knot Variance* needs to be checked, and they involve both an *invariance argument* and a *well-foundedness* argument.