From Natural Projection to Partial Model Checking and Back (Technical proofs and experiments)

Gabriele Costa[1⋆], David Basin[2], Chiara Bodei[3], Pierpaolo Degano[3], and Letterio Galletta[3⋆]

[1] DIBRIS, Università degli Studi di Genova
gabriele.costa@unige.it
[2] Department of Computer Science, ETH Zurich
basin@inf.ethz.ch
[3] Dipartimento di Informatica, Università di Pisa
{chiara,degano,galletta}@di.unipi.it

Overview

This document contains the appendix of the paper:

> Gabriele Costa, David Basin, Chiara Bodei, Pierpaolo Degano, and Letterio Galletta *"From Natural Projection to Partial Model Checking and Back"*. TACAS 2018 (accepted).

Technical Proofs

Here we provide the proofs of the theoretical results of the paper.

Definition 1. *A labeled transition system (LTS) is a tuple $A = (S_A, \Sigma_A, \rightarrow_A, \imath_A)$ where S_A is a finite set of states (with $\imath_A$ the initial state), Σ_A is a finite set of action labels, and $\rightarrow_A$: $S_A \times \Sigma_A \rightarrow S_A$ is the transition function. We write $t = s \xrightarrow{a} s'$ to denote a* transition*, whenever $\rightarrow_A (a, s) = s'$, and we call the state s the* source*, the symbol a the* action*, and the state s' the* destination.

A trace *$\sigma \in \mathcal{T}$ of an LTS A is either a single state s or a sequence of transitions $t_1 \cdot t_2 \cdot \ldots$ such that for each t_i, its destination is the source of t_{i+1} (if any). When unnecessary, we omit the source of t_{i+1}, and write a trace as the sequence $\sigma = s_0\mathbf{a_1}s_1\mathbf{a_2}s_2 \ldots \mathbf{a_n}s_n$, alternating elements of S_A and Σ_A (in boldface for readability). Finally, we denote by $[\![A, s]\!]$ the set of traces of A starting from state s and we write $[\![A]\!]$ for $[\![A, \imath_A]\!]$, i.e., for those traces starting from the initial state $\imath_A$.* □

⋆ This author is now affiliated with IMT Institute for Advanced Studies Lucca.

Definition 2. *Given two LTSs A and B such that $\Sigma_A \cap \Sigma_B = \Gamma$, the* synchronous product *of A and B is $A \parallel B = (S_A \times S_B, \Sigma_A \cup \Sigma_B, \rightarrow_{A\parallel B}, \langle \imath_A, \imath_B \rangle)$, where $\rightarrow_{A\parallel B}$ is as follows:*

$$\begin{array}{l}
\langle s_A, s_B\rangle \xrightarrow{a}_{A\parallel B} \langle s'_A, s_B\rangle \text{ if } s_A \xrightarrow{a}_A s'_A \text{ and } a \in \Sigma_A \setminus \Gamma \\
\langle s_A, s_B\rangle \xrightarrow{b}_{A\parallel B} \langle s_A, s'_B\rangle \text{ if } s_B \xrightarrow{b}_B s'_B \text{ and } b \in \Sigma_B \setminus \Gamma \\
\langle s_A, s_B\rangle \xrightarrow{\gamma}_{A\parallel B} \langle s'_A, s'_B\rangle \text{ if } s_A \xrightarrow{\gamma}_A s'_A, s_B \xrightarrow{\gamma}_A s'_B, \text{ and } \gamma \in \Gamma.
\end{array}$$
□

Definition 3. *For LTSs A and B with $\Gamma = \Sigma_A \cap \Sigma_B$, the* natural projection on A *of a trace σ, in symbols $P_A(\sigma)$, is defined as follows:*

$$\begin{array}{lll}
P_A(\langle s_A, s_B\rangle) & = s_A & \\
P_A(\langle s_A, s_B\rangle \xrightarrow{a}_{A\parallel B} \langle s'_A, s'_B\rangle \cdot \sigma) & = s_A \xrightarrow{a}_A s'_A \cdot P_A(\sigma) & \text{if } a \in \Sigma_A \\
P_A(\langle s_A, s_B\rangle \xrightarrow{b}_{A\parallel B} \langle s_A, s'_B\rangle \cdot \sigma) & = P_A(\sigma) & \text{if } b \in \Sigma_B \setminus \Gamma.
\end{array}$$

Natural projection on second component B is analogously defined. We extend the natural projection to sets of traces in the usual way: $P_A(T) = \{P_A(\sigma) \mid \sigma \in T\}$.

The inverse projection *of a trace σ over an LTS $A \parallel B$, in symbols $P_A^{-1}(\sigma)$, is defined as $P_A^{-1}(\sigma) = \{\sigma' \mid P_A(\sigma') = \sigma\}$. Its extension to sets is $P_A^{-1}(T) = \bigcup_{\sigma \in T} P_A^{-1}(\sigma)$.* □

Fact 1. $\qquad P_A([\![A \parallel B]\!]) \subseteq [\![A]\!] \qquad$ *and* $\qquad [\![A \parallel B]\!] = P_B^{-1}([\![A]\!]) \cap P_A^{-1}([\![B]\!])$.

Definition 4 (Syntax of the μ-calculus). *Given a set of variables $x \in X$ and an alphabet of actions $a \in \Sigma$,* assertions *$\varphi, \varphi' \in \mathcal{A}$ are given by the syntax:*

$$\varphi ::= \mathit{ff} \mid \mathit{tt} \mid x \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid [a]\varphi \mid \langle a\rangle\varphi.$$

An equation *is $x =_\pi \varphi$, where $\pi \in \{\mu, \nu\}$, μ denotes a minimum fixed point equation, and ν a maximum one. An* equation system *Φ is a possibly empty sequence (ϵ) of equations, where each variable x occurs in the left-hand side of at most single equation. Thus Φ is given by*

$$\Phi ::= x =_\pi \varphi; \Phi \mid \epsilon.$$

A top assertion *$\Phi{\downarrow}x$ projects the simultaneous solution of an equation system Φ onto the top variable x.* □

Definition 5 (Semantics of the μ-calculus [?]). *Let A be an LTS, and $\rho : X \to 2^{S_A}$ be an* environment *that maps variables to sets of A's states. Given an assertion φ, the* state semantics *of φ is the mapping $\|\cdot\| : \mathcal{A} \to (X \to 2^{S_A}) \to 2^{S_A}$ inductively defined as follows.*

$$\|\mathit{ff}\|_\rho = \emptyset \qquad \|\mathit{tt}\|_\rho = S_A \qquad \|x\|_\rho = \rho(x)$$

$$\|\varphi \wedge \varphi'\|_\rho = \|\varphi\|_\rho \cap \|\varphi'\|_\rho \qquad \|[a]\varphi\|_\rho = \{s \in S_A \mid \forall s'. s \xrightarrow{a}_A s' \Rightarrow s' \in \|\varphi\|_\rho\}$$

$$\|\varphi \vee \varphi'\|_\rho = \|\varphi\|_\rho \cup \|\varphi'\|_\rho \qquad \|\langle a\rangle\varphi\|_\rho = \{s \in S_A \mid \exists s'. s \xrightarrow{a}_A s' \wedge s' \in \|\varphi\|_\rho\}$$

We extend the state semantics from assertions to equation systems. First we introduce some auxiliary notation. The empty mapping is represented by $[\,]$, $[x \mapsto U]$ *is the environment where* U *is assigned to* x, *and* $\rho \circ \rho'$ *is the mapping obtained by composing* ρ *and* ρ'. *Given a function* $f(U)$ *on the powerset of* S_A, *let* $\pi U.f(U)$ *be the corresponding fixed-point. We now define the semantics of equation systems by:*

$$\begin{aligned}
&\|\epsilon\|_\rho &&= [\,] \\
&\|x =_\pi \varphi; \Phi\|_\rho &&= R(U^*) \quad \textit{where } U^* = \pi U.\|\varphi\|_{\rho \circ R(U)} \\
& && \qquad\qquad\;\; \textit{and } \; R(U) = [x \mapsto U] \circ \|\Phi\|_{\rho \circ [x \mapsto U]}.
\end{aligned}$$

Finally, for top assertions, let $\|\Phi \downarrow x\|$ *be a shorthand for* $\|\Phi\|_{[\,]}(x)$. □

Definition 6. *Given an LTS* A, *an environment* ρ, *and a state* $s \in S_A$, *the* trace semantics *of an assertion* φ *is a function* $\langle\!\langle \cdot \rangle\!\rangle : \mathcal{A} \to S_A \to (X \to 2^{S_A}) \to \mathcal{T}$, *that we also extend to equation systems, defined as follows.*

$$\langle\!\langle \varphi \rangle\!\rangle^s_\rho = \begin{cases} [\![A, s]\!] \text{ if } s \in \|\varphi\|_\rho \\ \emptyset \text{ otherwise} \end{cases} \qquad\qquad \langle\!\langle \Phi \rangle\!\rangle_\rho = \lambda x. \bigcup_{s \in \|\Phi\|_\rho(x)} [\![A, s]\!].$$

We write $\langle\!\langle \Phi\!\downarrow\! x \rangle\!\rangle$ *in place of* $\langle\!\langle \Phi \rangle\!\rangle_{[\,]}(x)$. □

Definition 7. *An LTS* A *satisfies a top assertion* $\Phi\!\downarrow\! x$, *in symbols* $A \models_s \Phi\!\downarrow\! x$, *if and only if* $\imath_A \in \|\Phi\!\downarrow\! x\|$. *Moreover, let* $A \models_\sigma \Phi \downarrow x$ *if and only if* $[\![A]\!] \subseteq \langle\!\langle \Phi\!\downarrow\! x \rangle\!\rangle$. □

Fact 2. $A \models_s \Phi\!\downarrow\! x$ *if and only if* $A \models_\sigma \Phi\!\downarrow\! x$.

Definition 8. *Given a top assertion* $\Phi \downarrow x$, *we define the quotienting of the assertion on an LTS* A *with respect to an alphabet* Σ_B *as follows.*

$$\Phi\!\downarrow\! x /\!\!/_{\Sigma_B} A = (\Phi /\!\!/_{\Sigma_B} A)\!\downarrow\! x_{\imath_A}, \;\; \textit{where}$$

$$\epsilon /\!\!/_{\Sigma_B} A = \epsilon \qquad (x =_\pi \varphi; \Phi) /\!\!/_{\Sigma_B} A = \begin{cases} x_{s_1} =_\pi \varphi /\!\!/_{\Sigma_B} s_1 \\ \vdots \\ x_{s_n} =_\pi \varphi /\!\!/_{\Sigma_B} s_n \end{cases} ; \; \Phi /\!\!/_{\Sigma_B} A \quad (\forall\; s_i \in S_A)$$

$$x /\!\!/_{\Sigma_B} s = x_s \qquad tt /\!\!/_{\Sigma_B} s = tt \qquad f\!f /\!\!/_{\Sigma_B} s = f\!f$$

$$\varphi \vee \varphi' /\!\!/_{\Sigma_B} s = \varphi /\!\!/_{\Sigma_B} s \vee \varphi' /\!\!/_{\Sigma_B} s \qquad \varphi \wedge \varphi' /\!\!/_{\Sigma_B} s = \varphi /\!\!/_{\Sigma_B} s \wedge \varphi' /\!\!/_{\Sigma_B} s$$

$$\begin{aligned}
&(\langle a \rangle \varphi) /\!\!/_{\Sigma_B} s = \bigvee_{s \xrightarrow{a} s'} \varphi /\!\!/_{\Sigma_B} s' &&([a]\varphi) /\!\!/_{\Sigma_B} s = \bigwedge_{s \xrightarrow{a} s'} \varphi /\!\!/_{\Sigma_B} s' &&\textit{if } a \in \Sigma_A \setminus \Gamma \\
&(\langle b \rangle \varphi) /\!\!/_{\Sigma_B} s = \langle b \rangle (\varphi /\!\!/_{\Sigma_B} s) &&([b]\varphi) /\!\!/_{\Sigma_B} s = [b](\varphi /\!\!/_{\Sigma_B} s) &&\textit{if } b \in \Sigma_B \setminus \Gamma \\
&(\langle \gamma \rangle \varphi) /\!\!/_{\Sigma_B} s = \bigvee_{s \xrightarrow{\gamma} s'} \langle \gamma \rangle (\varphi /\!\!/_{\Sigma_B} s') &&([\gamma]\varphi) /\!\!/_{\Sigma_B} s = \bigwedge_{s \xrightarrow{\gamma} s'} [\gamma](\varphi /\!\!/_{\Sigma_B} s') &&\textit{if } \gamma \in \Gamma.
\end{aligned}$$

□

Definition 9. *Given a synchronous product $A \parallel B$, we define* $\Delta_B(\cdot) : (X \to 2^{S_A \times S_B}) \to (X_{S_A} \to 2^{S_B})$ *as*

$$\Delta_B(\rho) = \bigoplus_{x \in Dom(\rho)} \bigoplus_{s_A \in S_A} [x_{s_A} \mapsto U^x_B(s_A)], \textit{ where } U^x_B(s_A) = \{s_B \mid \langle s_A, s_B \rangle \in \rho(x)\} \quad \square$$

Lemma 1. *For all A, B, ρ and φ on $A \parallel B$, $\langle s_A, s_B \rangle \in \|\varphi\|_\rho \iff s_B \in \|\varphi /\!\!/_{\Sigma_B} s_A\|_{\Delta_B(\rho)}$.*

Proof. By induction over the structure of φ.

- Cases tt and $f\!f$. Trivial.
- Case x. By definition of $\Delta_B(\rho)$.
- Cases $\varphi \wedge \varphi'$, $\varphi \vee \varphi'$. By the induction hypothesis.
- Case $\langle a \rangle \varphi$. By the Definition 5, $\langle s_A, s_B \rangle \in \|\langle a \rangle \varphi\|_\rho$ if and only if $\exists s'_A, s'_B$ such that $\langle s_A, s_B \rangle \xrightarrow{a}_{A\parallel B} \langle s'_A, s'_B \rangle \wedge \langle s'_A, s'_B \rangle \in \|\varphi\|_\rho$. By the induction hypothesis, this is equivalent to

$$\exists s'_A, s'_B \text{ such that } \langle s_A, s_B \rangle \xrightarrow{a}_{A\parallel B} \langle s'_A, s'_B \rangle \wedge s'_B \in \|\varphi /\!\!/_{\Sigma_B} s'_A\|_{\Delta_B(\rho)} \qquad (1)$$

  Then we consider three exhaustive cases.
  - $a \in \Sigma_A \setminus \Gamma$. Here $s'_B = s_B$ and (1) is satisfied if and only if $s_B \in \| \bigvee_{s_A \xrightarrow{a}_A s'} \varphi /\!\!/_{\Sigma_B} s' \|_{\Delta_B(\rho)}$. We conclude by applying Definition 8.
  - $a \in \Sigma_B \setminus \Gamma$. In this case $s'_A = s_A$ and, by Definition 5, (1) is equivalent to claiming $s_B \in \|\langle a \rangle (\varphi /\!\!/_{\Sigma_B} s_A)\|_{\Delta_B(\rho)}$. Again, we close the case by applying Definition 8.
  - $a \in \Gamma$. We combine the reasoning of the two previous cases to conclude that $s_B \in \| \bigvee_{s_A \xrightarrow{a}_A s'} \langle a \rangle (\varphi /\!\!/_B s') \|_{\Delta_{\Sigma_B}(\rho)}$.
- Case $[a]\varphi$. Symmetric to the previous one.

$\square$

Lemma 2. *For all A, B, ρ and Φ on $A \parallel B$, $\Delta_B(\|\Phi\|_\rho) = \|\Phi /\!\!/_{\Sigma_B} A\|_{\Delta_B(\rho)}$.*

Proof. We proceed by induction on the structure of Φ.

- Base case: $\Phi = \epsilon$. Trivial.
- Inductive step: $\Phi = x =_\pi \varphi; \Phi'$. By definition $\|\Phi\|_\rho = [x \mapsto U^*] \circ \|\Phi'\|_{\rho \circ [x \mapsto U^*]}$ where U^* is the fixed point computed according to Definition 5. Thus, we have that $\Delta_B(\|\Phi\|_\rho) = \Delta_B([x \mapsto U^*] \circ \|\Phi'\|_{\rho \circ [x \mapsto U^*]}) = \Delta_B([x \mapsto U^*]) \circ \Delta_B(\|\Phi'\|_{\rho \circ [x \mapsto U^*]})$. By the induction hypothesis this reduces to

$$\Delta_B([x \mapsto U^*]) \circ \|\Phi' /\!\!/_B A\|_{\Delta_B(\rho) \circ \Delta_B([x \mapsto U^*])}. \qquad (2)$$

By Definition 9, $\Delta_B([x \mapsto U^*]) = \bigoplus_{s \in S_A} [x_s \mapsto U^*_{B,s}]$ where $U^*_{B,s} = \{s' \mid \langle s, s' \rangle \in U^*\}$. By replacing U^* with its definition we obtain that

$$U^*_{B,s} = \{s' \mid \langle s, s' \rangle \in \pi U. \|\varphi\|_{\rho \circ R(U)}\},$$

which we rewrite it to

$$U^*_{B,s} = \pi U. \{s' \mid \langle s, s' \rangle \in \|\varphi\|_{\rho \circ R(U)}\}.$$

By Lemma 1, this is equivalent to

$$U^*_{B,s} = \pi U_{B,s}. \|\varphi /\!\!/_{\Sigma_B} s\|_{\Delta_B(\rho) \circ \Delta_B(R(U))},$$

where $\Delta_B(R(U)) = \Delta_B([x \mapsto U] \circ \|\Phi'\|_{\rho \circ [x \mapsto U]})$. By Definition 9, we have $\bigoplus_{s \in S_A} [x_s \mapsto U_{B,s}] \circ \Delta_B(\|\Phi'\|_{\rho \circ [x \mapsto U]}) = \bigoplus_{s \in S_A} [x_s \mapsto U_{B,s}] \circ \|\Phi' /\!\!/_{\Sigma_B} A\|_{\Delta_B(\rho) \circ \bigoplus_{s \in S_A} [x_s \mapsto U_{B,s}]}$ (by the induction hypothesis).
As a consequence, we rewrite (2) to[4]

$$\bigoplus_{s \in S_A} [x_s \mapsto U^*_{B,s}] \circ \|\Phi' /\!\!/_{\Sigma_B} A\|_{\Delta_B(\rho) \circ \bigoplus_{s \in S_A} [x_s \mapsto U^*_{B,s}]},$$

which, after repeatedly applying Definition 5 to each element $s \in S_A$ turns out to be $\|\Phi /\!\!/_{\Sigma_B} A\|_{\Delta_B(\rho)}$.

□

Corollary 1. *For all A, B, ρ, x and Φ on $A \parallel B$,*

$$\langle s_A, s_B \rangle \in \|\Phi\|_\rho(x) \iff s_B \in \|\Phi /\!\!/_{\Sigma_B} A\|_{\Delta_B(\rho)}(x_{s_A}).$$

Theorem 3. *For all A, B, x and Φ on $A \parallel B$, $\langle\!\langle \Phi \downarrow x /\!\!/_{\Sigma_B} A \rangle\!\rangle = P_B(\langle\!\langle \Phi \downarrow x \rangle\!\rangle)$.*

Proof. By Definition 6, the thesis becomes

$$\langle\!\langle \Phi /\!\!/_{\Sigma_B} A \rangle\!\rangle_{[]}(x_{\imath_A}) = P_B(\langle\!\langle \Phi \rangle\!\rangle_{[]}(x)),$$

which holds if and only if

$$\langle \imath_A, \imath_B \rangle \in \|\Phi\|_{[]}(x) \iff \imath_B \in \|\Phi /\!\!/_{\Sigma_B} A\|_{[]}(x_{\imath_A}).$$

The thesis follows by Corollary 1. □

Theorem 4. *For all A, B, x and Φ on $A \parallel B$,*

$$A \parallel B \models_\varsigma \Phi \downarrow x \qquad (\varsigma \in \{s, \sigma\})$$

if and only if any of the following equivalent statements holds
1. $A \models_\varsigma \Phi \downarrow x /\!\!/_{\Sigma_A} B$ 2. $B \models_\varsigma \Phi \downarrow x /\!\!/_{\Sigma_B} A$
3. $A \models_\sigma P_A(\langle\!\langle \Phi \downarrow x \rangle\!\rangle)$ 4. $B \models_\sigma P_B(\langle\!\langle \Phi \downarrow x \rangle\!\rangle)$

Proof. The equivalence of items 1 and 2 and $A \parallel B \models_\varsigma \Phi \downarrow x$ is in [**?**] (with the additional use of Theorem 3). The other equivalences follow immediately by Theorem 3 (and by the commutativity of $\parallel$). □

[4] Notice that the order of the x_s equations is immaterial as they form a π-block.

```
Begin proc quotient
   input P = <$S_P$ , $\Sigma_P$ , $\rightarrow_P$ , $i_P$>
   input A = <$S_A$ , $\Sigma_A$ , $\rightarrow_A$ , $i_A$>
   input $\Sigma_B$

1:  $S := (S_P \times S_A) \setminus \bigcup_{a \in \Sigma_A} \{(s_P, r_A) \mid s_P \overset{a}{\not\rightarrow}_P \wedge r_A \xrightarrow{a}_A\}$
2:  $i := (i_P, i_A)$
3:  $\rightarrow := \bigcup_{s_P} \begin{cases} \bigcup_{a \in \Sigma_A \setminus \Gamma} \{((s_P, r_A), \lambda, (s'_P, r'_A)) \mid s_P \xrightarrow{a}_P s'_P \wedge r_A \xrightarrow{a}_A r'_A\} \\ \bigcup_{a \in \Sigma_B \setminus \Gamma} \{((s_P, r_A), a, (s'_P, r_A)) \mid s_P \xrightarrow{a}_P s'_P\} \\ \bigcup_{a \in \Gamma} \{((s_P, r_A), a, (s'_P, r'_A)) \mid s_P \xrightarrow{a}_P s'_P \wedge r_A \xrightarrow{a}_A r'_A\} \end{cases}$
4:  B := <$S$ , $\Sigma_B$ , $\rightarrow_S$ , $i$>
5:  output unify(B)
End proc
```

Table 1. The quotienting algorithm.

Correctness and complexity

Below we discuss the correctness and computational complexity of the algorithms presented in the paper.

Correctness

To prove the correctness of the algorithm, we show that it is equivalent to the standard quotienting operator applied to a suitable encoding of an LTS as a specification of the equational μ-calculus.

Encoding. Given an LTS $A = \langle S, \Sigma, \rightarrow, s_i \rangle$ we build a system of equations as follows:

$$\Phi_A = \{x^{s_1} =_\mu \mathcal{A}(s_1); \ldots ; x^{s_n} =_\mu \mathcal{A}(s_n)\}$$

where $S = \{s_1, \ldots, s_n\}$ and $\mathcal{A}(s) = \bigwedge_{s \overset{b}{\not\rightarrow}} [b]\mathit{ff} \;\wedge \bigwedge_{s \xrightarrow{a} s'} [a]x^{s'}$

Thus we define the top assertion of the formula derived from A as $\Phi_A \downarrow x^{s_i}$.

Quotienting. Starting from the inputs of `quotient`, we now evaluate $\Phi_P \downarrow x^{i_P} /\!\!/_{\Sigma_B} A$. The resulting equations system is

$$\Phi' = \begin{cases} x^{s_1}_{r_1} =_\mu \mathcal{A}(s_1) /\!\!/_{\Sigma_B} r_1 \\ \quad\cdots \\ x^{s_i}_{r_j} =_\mu \mathcal{A}(s_i) /\!\!/_{\Sigma_B} r_j \\ \quad\cdots \\ x^{s_n}_{r_m} =_\mu \mathcal{A}(s_n) /\!\!/_{\Sigma_B} r_m \end{cases}$$

```
Begin proc unify
  input B = <$S_B$, $\Sigma_B$, $\to_B$, $i_B$>

  $I$ := $\lambda$-close({$i_B$})
  $R$, $S$ := {$I$}
  $\to$ := $\emptyset$
  while $S \neq \emptyset$ do
    $Q$ := pick&remove($S$)
    for each $a \in \Sigma_B \setminus \{\lambda\}$
      $M$ := $\wedge$-move($Q$, $a$)
      if $M \neq \emptyset$ then
        $\to$ := $\to \cup \{(Q, a, M)\}$
        if $M \notin R$ then $S$ := $S \cup \{M\}$; $R$ := $R \cup \{M\}$ end if
      end if
    end for
  end while
  output <$R$, $\Sigma_B \setminus \{\lambda\}$, $\to$, $I$>
end proc
```

```
Begin proc ∧-move
  input $Q$
  input $a$

  $M$ := $\lambda$-close($\bigcap^{q \in Q} \{q' | q \xrightarrow{a}_B q'\}$)
  output $M$
end proc
```

Table 2. The unification algorithm.

with all the equations of the following form, where $\alpha \in \Sigma_A \setminus \Gamma$, $\beta \in \Sigma_B \setminus \Gamma$ and $\gamma \in \Gamma$.

$$x^{s_i}_{r_j} =_\mu \overbrace{\bigwedge_{\substack{s_i \not\xrightarrow{a} \\ r_j \xrightarrow{a} r'}} ff}^{(1)} \wedge \overbrace{\bigwedge_{\substack{s_i \not\xrightarrow{a} \\ r_j \not\xrightarrow{a}}} [a]ff}^{(2)} \wedge \overbrace{\bigwedge_{\substack{s_i \xrightarrow{\alpha} s' \\ r_j \xrightarrow{\alpha} r'}} x^{s'}_{r'}}^{(3)} \wedge \overbrace{\bigwedge_{s_i \xrightarrow{\beta} s'} [\beta] x^{s'}_{r_j}}^{(4)} \wedge \overbrace{\bigwedge_{\substack{s_i \xrightarrow{\gamma} s' \\ r_j \xrightarrow{\gamma} r'}} [\gamma] x^{s'}_{r'}}^{(5)}$$

Trivially, the equation system described above corresponds to the non-deterministic transition system obtained by the `quotient` algorithm. A state (s, r) results in a variable associated to an assertion that characterizes the outgoing transitions distinguishing among (1) a is required but not done, (2) a is not allowed, (3) λ moves, (4) Σ_B, and (5) Γ actions.

Correctness. To conclude we show that all the steps of the algorithms described above correspond to valid transformations, i.e., they preserve equivalence. A detailed description of the first two transformations can be found in [**?**].

- *Constant propagation* is applied to remove equations of the form $x =_\mu ff$. This step is carried out by the `quotient` algorithm when removing the corresponding states from the transition system.
- *Unguardedness removal* carries out the following transformation.

$$\begin{cases} x =_\mu \varphi \\ \vdots \\ y =_\mu \varphi' \end{cases} \quad \text{becomes} \quad \begin{cases} x =_\mu \varphi\{\varphi'/x\} \\ \vdots \\ y =_\mu \varphi' \end{cases}$$

All the occurrences of y in the first equation are replaced with the assertion associated to y. Notice that this transformation only applies if all the occurrences of y are unguarded, i.e., not under the scope of any modal operator, in the first equation. Also, we extend it to remove redundant recurrences, namely we transform $x =_\mu \varphi \wedge x$ in $x =_\mu \varphi$. In our algorithm, this operation corresponds to a λ-closure.

– *Variable introduction* requires more attention. It is simple to verify that the previous transformations do not preserve the structure of our equations. Indeed, unguardedness removal can introduce in the assertions more instances of the same action modality, while we require exactly one. Concretely, the assertions have the form

$$x =_\mu [a]v_1^a \wedge \ldots \wedge [a]v_k^a \wedge [b]v_1^b \cdots ,$$

where v stands for either a variable or ff. If some of the v_i^a are equal to ff, $[a](v_1^a \wedge \ldots \wedge v_k^a)$ reduces to $[a]\mathit{ff}$. Otherwise, we rewrite it as $x =_\mu [a](y_1 \wedge \ldots \wedge y_k) \wedge [b](\ldots)$. Thus, we replace the conjunctions of variables with $[a](y_{\{1,\ldots,k\}})$ and we introduce a new equation $y_{\{1,\ldots,k\}} =_\mu y_1 \wedge \ldots \wedge y_k$. Clearly, the number of these new variables is bounded by $2^{|S|}$. This transformation, plus unguardedness removal, corresponds to the $\wedge$-`move` operation. It is simple to notice that these transformations restore the format of our encoding, thus denoting an LTS that is the output of our algorithm.

Complexity

We estimate the worst case complexity of the `quotient` algorithm. For simplicity, we assume that $|\Gamma| = |\Sigma_A \setminus \Gamma| = |\Sigma_B \setminus \Gamma| = m$ and $|S_A| = |S_P| = n$. The first part, i.e., the generation of the non-deterministic transition system, requires at most $|\rightarrow_P| \cdot |\rightarrow_A| \leq m^2 n^4$ steps. The transition system has at most n^2 states.

Concerning `unify`, we first observe the following facts. The algorithm works on the λ-closures of the states of the transition system. Similarly to the ε-closures of an NFA, they can be computed in advance (see [**?**]). The cost is cubic with respect to the number of states, i.e., $O(n^6)$ in our case. The total number of closures is bounded by n^2.

At each step, $\wedge$-`move` computes the sets of the a-reachable states, starting from one of the closures (which has size at most n^2). Since there are no more than nm Γ-transitions (and at most m Σ_B-transitions), in $n^3 m$ steps we obtain the set on which we compute the λ-closure. Since we already computed them, we just need to combine the existing ones.

To conclude we observe that $\wedge$-`move` iterates at most $n^2 m$ times. In particular, new elements are added to S whenever a new λ-closure is generated. However, these closures are computed from a set of states that are a-reachable from all the states of another closure. Clearly, there can be at most $n^2 m$ distinct closures of this type, otherwise there would be more than $n^4 m^2$ edges in the transition system. The overall complexity is then $O(2n^5 m^3 + n^6 + n^4 m^2)$.

n	States	Transitions	Time (ms)
5	24	36	1
10	74	121	6
15	149	256	50
20	249	441	347
25	374	676	1205
30	524	961	3729
35	699	1296	9144
40	899	1681	21768
45	1124	2116	43750
50	1374	2601	58869
55	1649	3136	97184
60	1949	3721	197555
65	2274	4356	307549
70	2624	5041	462417
75	2999	5776	674102
80	3399	6561	1094193
85	3824	7396	1507860
90	4274	8281	2070366
95	4749	9216	2796649
100	5249	10201	5685985

n	States	Transitions	Time (ms)
5	54	54	30
10	209	209	88
15	464	464	398
20	819	819	1979
25	1274	1274	8013
30	1829	1829	23953
35	2484	2484	84634
40	3239	3239	153595
45	4094	4094	342434
50	5049	5049	194525
55	6104	6104	325718
60	7259	7259	624646
65	8514	8514	959671
70	9869	9869	1487936
75	11324	11324	2429919
80	12879	12879	3888799
85	14534	14534	5546023
90	16289	16289	7353658
95	18144	18144	9589415
100	20099	20099	12922962

Table 3. The dimension of some instances of the SCP problem (on the left) and CSP problem (on the right) and the time required by our PESTS to solve them.

Experiments

We tested our tool on the case studies presented in the running example of the paper and we used our tool PESTS to solve the Submodule Construction Problem (SCP) and the Controller Synthesis Problem (CSP).

In Table 3, we report the results of these experiments, carried on an Intel Core i7-6700HQ, CPU 2.60GHz, with 16 GB RAM. The first column represents the buffer size n (and the UAV capacity is $n-1$); the second and third column represent the number of states and transitions of the automata manipulated by the algorithm (recall from the example in sub-section 4.2 that the automata of CSP have the same number of states and transitions); and the last column represents the time necessary to compute the solution.

Although PESTS is at the early stages of development the results we obtained are quite promising. We believe that a suitable engineering of the tool can improve its performance and we plan to carry out a comparison with other existing tools as part of the future work.