Compositionality and locality for improving model checking in the selective mu-calculus

Antonella Santone, Gigliola Vaglini

Università del Sannio, Facoltà di Ingegneria, Corso Garibaldi 107, Palazzo Bosco Lucarelli, 82100 Benevento, Italy
Dipartimento di Ingegneria dell’Informazione, Università di Pisa, Via Diotisalvi, 2, I-56126 Pisa, Italy

Received 7 January 2004; received in revised form 25 March 2004; accepted 2 July 2004
Available online 8 September 2004

Abstract

Model checking is an automatic technique for verifying properties of finite concurrent systems on a structure that represents the states of the system; the crucial point of the technique is to avoid the computation of all the possible states. In this paper a method of proof for concurrent systems is presented that combines several approaches to meet the previous goal. The method exploits compositionality issues, in the presence of a parallel composition of processes, to compute at most the states of each sequential process, and not their combinations; moreover the method employs abstraction techniques to compute but a subset of the states of each sequential process. Finally, tableau-based proofs are used to allow the dynamic generation of the system states when needed, taking into account the goal of the formula verification. The tableau system is proved finite, sound and complete, for finite state systems.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Software engineering; Model checking; Temporal logic; Compositionality; Tableau system
1. Introduction

When model checking properties of finite concurrent systems, the verification of properties expressed by means of logic formulae is carried on a structure, for instance a transition system, representing the possible states of the system. Obviously, it is not possible to compute all states of any system, even if it is finite. To overcome this problem different methods can be taken into account. First of all, the size of the state space explodes because parallelism is generally modeled by means of interleaving, thus techniques based on compositionality aid in reducing this space [2,1,14]: the traces of the states of the processes composed in parallel are kept separate to store at most the states of each process, and not their combinations. Reduction in the state space of a system can be also obtained computing but a subset of the states of a concurrent system by means of abstraction techniques (see for example [5,7,8]). The temporal logic called selective mu-calculus, equi-expressive to propositional mu-calculus [12,15], was introduced [3,4] for this purpose: the automatic construction of an abstraction of the state space is directly induced by each formula that (syntactically) specifies only the actions whose occurrence can alter its truth-value. Good results can be also obtained by exploiting proof techniques which do not require that every state in the system is computed before verification: each state is generated, in order to determine if the system has a property, only if the property itself renders such generation necessary. An example of the last approach is the so called local model checking [15,10,16].

This paper presents a method of proof for systems described by CCS processes (actually, we use a slightly modified version of CCS) that combines the three previously sketched approaches. Following compositional techniques, the method takes into account the structure of the process describing the system and computes separate sets of states for each process in a parallel composition. Moreover the method takes into account only a subset of the states of each sub-process by exploiting the potentialities of the selective mu-calculus. Finally, the method relies on the construction of tableau-based proofs and, when possible, the exhaustive traversal of the states of the system is avoided. More precisely, consider two sequential processes composed in parallel: the tableau rules act on pairs of sets of states, each set concerning a sequential process. The global move of the concurrent process is simulated by separate moves of the component sequential processes through equal communication patterns. The tableau system is finite, sound and complete, for finite state systems.

The paper is organized as follows. Section 2 contains a short review of CCS and of the selective mu-calculus; Section 3 presents the tableau system. Section 4 discusses the advantages of the presented approach in terms of space complexity; while Section 5 discusses the approach with respect to some related works and presents the conclusions.

2. Background

2.1. CCS

We assume the reader to be familiar with CCS [13], a process algebra widely used in the specification of concurrent and distributed systems, and we recall only some basic definitions. The reader can refer to [13] for more details.
A process is the composition of the atomic actions belonging to a finite set \( \mathcal{A} = \{a, b, \ldots \} \) by means of a set of operators. For the purposes of this work and without loss of generality, we consider a class of CCS processes of the form produced by the following syntax; we call this class Flat-CCS (FCCS):

\[
\begin{align*}
\mathit{nil} &::= p \parallel \mathit{nil} \mid q \\
q &::= \mathit{nil} \mid x \parallel a.q \mid q + q \mid q[f]
\end{align*}
\]

where \( x \) ranges over \( \mathcal{A} \), \( C \subseteq \mathcal{A} \), \( f : \mathcal{A} \to \mathcal{A} \) is a relabeling function and \( x \) ranges over a set of constant names. Each constant \( x \) is defined by a constant definition \( x = q \). \( L(p) \) denotes the sort of the process \( p \); more precisely, the sort of the FCCS process \( p \) is the set \( L(p) \subseteq \mathcal{A} \) that is the least solution of the following recursive definition.

\[
\begin{align*}
L(\mathit{nil}) &= \emptyset \\
L(a.p) &= L(p) \cup \{a\} \\
L(p + q) &= L(p) \cup L(q) \\
L(p \parallel q) &= L(p) \cup L(q) \\
L(p[f]) &= \{f(\alpha) : \alpha \in L(p)\} \\
L(x) &= L(p) \text{ if } x = p.
\end{align*}
\]

When actions are composed only by means of the operators action prefix \( (a.q) \), choice \( (q_1 + q_2) \) and relabeling \( (q[f]) \), we obtain a sequential process. On the other hand, when sequential processes are composed by means of the parallel operator \( (q_1 \parallel q_2) \), we obtain a parallel process; \( C \) denotes the communication actions, i.e., the actions that occur only if both partners are ready to perform them. For simplicity reasons, we shall write \( q_1 \parallel q_2 \) when \( C = L(q_1) \cap L(q_2) \). We denote by \( \mathcal{P} \) the set of FCCS processes.

The structural operational semantics of a process \( p \) is given by the relation \( \longrightarrow \) which is the minimal relation defined by the rules in Table 1. The same set of conditional rules describes also the transition relation \( \longrightarrow \) of the automaton corresponding to the process. This automaton, called the transition system of \( p \), is denoted by \( S(p) = (S, \mathcal{A}, \longrightarrow, s_p) \), where \( S \) is a set of states, and \( s_p \in S \) is the initial state of \( S(p) \). With abuse of notation, hereafter we shall use the same symbol \( \longrightarrow \) for denoting both the operational semantics and the transition relation among the states of the transition system: i.e., we write \( p \xrightarrow{\alpha} q \), both for \( p, q \in \mathcal{P} \), and for \( p, q \in S \). If \( \delta \in \mathcal{A}^* \) and \( \delta = \alpha_1 \ldots \alpha_n \), we write \( p \xrightarrow{\delta} q \) for \( p \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} q \). In Table 1 the symmetrical rules for choice and parallel composition are not shown; moreover, differently from standard CCS, when both \( q_1 \) and \( q_2 \) are able to perform the action \( \alpha \), \( q_1 \parallel q_2 \) performs the action \( \alpha \) too (see rule \textbf{Com} in Table 1). An FCCS process is finite if its transition system is finite. From now on, we consider only finite processes.

**Example 1.** The FCCS process \( U \parallel M \) in Table 2 describes a tea dispenser (\( C = \{\text{coin, tea, sugar, lemon, help, go}\} \)). After the user (\( U \)) inserts a \text{coin}, the machine (\( M \)) shows one of the following messages on the display: \text{all}, if both sugar and lemon are
Table 1
Structural operational semantics of FCCS

<table>
<thead>
<tr>
<th>Act</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha. q \xrightarrow{a} s q )</td>
<td>( q_1 \xrightarrow{\alpha} s q'_1 )</td>
</tr>
<tr>
<td>( q \xrightarrow{a} s q' )</td>
<td>( q_1 \xrightarrow{\alpha} s q'_1 )</td>
</tr>
<tr>
<td>Con</td>
<td>Par</td>
</tr>
<tr>
<td>( x \xrightarrow{a} s q' )</td>
<td>( q_1 \xrightarrow{\alpha} s q'_1 )</td>
</tr>
<tr>
<td>( q_1 \xrightarrow{\alpha} s q'_1, q_2 \xrightarrow{a} s q'_2 )</td>
<td>( q \xrightarrow{a} s q' )</td>
</tr>
<tr>
<td>Rel</td>
<td></td>
</tr>
</tbody>
</table>
| \( q_1 \parallel c q_2 \xrightarrow{a} s q'_1 \parallel c q_2 \) | \( q[f] \xrightarrow{a} s q'[f] \)

available, only_lemon (only_sugar), if only the lemon (sugar) is available, nothing, if neither sugar nor lemon are available. The user may read the message on the display (reads), or not (no_reads), in both cases he/she is able to nondeterministically select any ingredient to be added to the cup of tea, however the machine forces the right choice (the pressing of the button lemon and/or sugar is registered only if the machine can supply the related ingredient). Nevertheless, the machine supplies also a help button to repeat the message for the user who has not read the previous message. After the user makes his/her choice, he/she can press the button go and collect the cup of tea, plus the spoon, if he/she has requested the sugar. The transition system \( S(U \parallel \sigma) \) has 43 states and 73 transitions.

2.2. Selective mu-calculus

In the model checking framework, systems can be represented by transition systems and requirements are expressed as formulae in a temporal logic. Thus model checkers accept two inputs, a transition system and a formula, and return “true” if the formula is satisfied and “false” otherwise. The major problem in this framework is the so-called state explosion: systems are often described by transition systems with a prohibitive number of states. In [3,4] the temporal logic selective mu-calculus was defined for reducing the state explosion problem. The reduction is driven by the syntactic structure of the formula. The selective mu-calculus is a variant of mu-calculus [12,15] and differs from it in the definition of the modal operators. The syntax of the logic is the following:

\[
\varphi ::= t \mid f \mid \varphi \mid \varphi_1 \lor \varphi_2 \mid \varphi_1 \land \varphi_2 \mid [K] \varphi \mid (K) \varphi \mid vZ.\varphi \mid \mu Z.\varphi
\]

where \( K \) and \( R \) range over sets of actions, while \( Z \) ranges over a set of variables.

As in standard mu-calculus, a fixed point formula has the form \( \mu Z.\varphi (vZ.\varphi) \), where \( \mu Z (vZ) \) binds free occurrences of \( Z \) in \( \varphi \). An occurrence of \( Z \) is free if it is not within the scope of a binder \( \mu Z (vZ) \). A formula is closed if it contains no free variables. \( \mu Z.\varphi \) is the least fix-point of the recursive equation \( Z = \varphi \), while \( vZ.\varphi \) is the greatest one. From now on, we consider only closed formulae.
Table 2
A tea dispenser

\[ M = \text{coin.} \text{(all.tea.} \text{(go.M + lemon.(sugar.go.spoon.M + go.M) + sugar.(lemon.go.spoon.M + go.spoon.M)) + only_lemon.tea.} \text{(lemon.} \text{(go.M + help.only_lemon.go.M) + go.M) + only_sugar.tea.} \text{(sugar.} \text{(go.spoon.M + help.only_sugar.go.M) + go.M) + nothing.tea.} \text{(go.M + help.nothing.go.M)) + only_lemon.tea.} \text{(lemon.} \text{(go.M + help.only_lemon.go.M) + go.M) + only_sugar.tea.} \text{(sugar.} \text{(go.spoon.M + help.only_sugar.go.M) + go.M) + nothing.tea.} \text{(go.M + help.nothing.go.M))} \]

\[ U = \text{coin.} \text{(no_reads.tea.} \text{(go.U + lemon.(sugar.go.u + help.go.u) + sugar.(lemon.go.u + help.go.u) + lemon.go.u + sugar.go.u + help.go.u) + reads.tea.} \text{(go.u + lemon.(sugar.go.u + go.u) + sugar.(lemon.go.u + go.u)))} \]

The modal operators have the following intuitive meaning:

\[[K]_R \varphi\] means that \( \varphi \) must be satisfied after the performance of each sequence of actions not belonging to \( R \cup K \) and followed by an action in \( K \).

\[\langle K \rangle_R \varphi\] means that at least a sequence of actions not belonging to \( R \cup K \) and followed by an action in \( K \) has to be performed, \( \varphi \) must be satisfied afterwards.

The selective \( \mu \)-calculus is equi-expressive to \( \mu \)-calculus; in fact, as shown in [4], selective formulae can be translated into \( \mu \)-calculus recursive formulae as follows.

\[ [K]_R \varphi = \nu Z. [K] \varphi \land [A \setminus (K \cup R)] Z \]
\[ \langle K \rangle_R \varphi = \mu Z. \langle K \rangle \varphi \lor (A \setminus (K \cup R)) Z. \]

On the other hand, \( \mu \)-calculus formulae can be translated into selective formulae as follows.

\[ [K] \varphi = [K]_A \varphi \]
\[ \langle K \rangle \varphi = \langle K \rangle_A \varphi. \]

The formal definition of satisfaction of the closed formula \( \varphi \) by the state \( p \) of a transition system, written \( p \models \varphi \), is given in Table 3. The process \( p \) satisfies \( \varphi \) if the initial state of \( S(p) \) satisfies \( \varphi \). In Table 3 the relation \( \Longrightarrow_I \subseteq S \times I \times S \) is used: such a relation is parametric with respect to \( I \subseteq A \) and ignores all “non-interesting actions” (i.e., those in \( A \setminus I \)).
Definition 1. Consider the transition system of the FCCS process \( p \), \( S(p) = (S, A, \rightarrow, s_p) \), and the set \( I \subseteq A \). For each \( \alpha \in I \), \( \gamma \in A^* \), and \( p, q \in S \), \( \Rightarrow I \subseteq S \times I \times S \) is such that

\[
p \overset{\alpha}{\Rightarrow} I q \overset{\text{def}}{=} p \overset{\gamma \alpha}{\Rightarrow} q \text{ and } \Pi_I(\gamma) = \epsilon.
\]

\( \Pi_I(\gamma) \) is the string over \( I \) obtained from \( \gamma \) by deleting all symbols not belonging to \( I \). Note that, if \( I = \emptyset \), \( \Pi_I(\gamma) = \epsilon \), for any \( \gamma \).

Informally, \( p \overset{\alpha}{\Rightarrow} I q \) means that it is possible to pass from \( p \) to \( q \) by performing a (possibly empty) sequence, \( \gamma \), of actions not belonging to \( I \) followed by the action \( \alpha \). Note that \( \Rightarrow A = \rightarrow \).

If \( \delta \in I^* \) and \( \delta = \alpha_1 \ldots \alpha_n \), we write \( p \overset{\delta}{\Rightarrow} I q \) for \( p \overset{\alpha_1}{\Rightarrow} \cdots \overset{\alpha_n}{\Rightarrow} I q \).

Example 2. Consider the following three FCCS processes:

\[
p_1 = c.a.b.p_1 + a.c.p_1
\]

\[
p_2 = b.c.a.p_2 + b.a.c.a.p_2
\]

\[
p_3 = a.b.p_3 + a.p_3.
\]

The transition systems of \( p_i \), with \( i \in [1..3] \), are shown in Fig. 2. Now, consider the following formulae:

\[
\psi_1 = [a]_{b}\jotff: \text{“it is not possible to perform the action } a \text{ without having performed the action } b \text{ before”}.
\]

\[
\psi_2 = \langle a \rangle_{\jottrue}: \text{“it is possible to perform the action } a \text{ after having performed any action”}.
\]

\[
\psi_3 = \nu Z. [a]_{b}(Z \wedge [a]_{[b,c]}\jotff): \text{“it always holds that, after } a \text{ has occurred, the successive } a \text{ can occur only after either the action } b \text{ or the action } c \text{”}.
\]

It holds that:

\[
p_1 \not\models \psi_1 \quad p_2 \models \psi_1 \quad p_3 \not\models \psi_1
\]

\[
p_1 \models \psi_2 \quad p_2 \models \psi_2 \quad p_3 \models \psi_2
\]

\[
p_1 \models \psi_3 \quad p_2 \not\models \psi_3 \quad p_3 \not\models \psi_3.
\]

Given the formula \( \varphi \), the set of its occurring actions, \( O(\varphi) \), is the union of all sets \( K \) and \( R \) used in the modal operators \( \langle [K]_R \rangle, \langle K \rangle_R \) in \( \varphi \). Given the process \( p \), the transition system \( S_{O(\varphi)}(p) = (S, O(\varphi), \Rightarrow O(\varphi), s_p) \) is an abstraction of \( S(p) \) that preserves the property \( \varphi \). In [4,3] it is stated that \( S(p) \) and \( S_{O(\varphi)}(p) \) have the same behavior with respect to \( O(\varphi) \). More precisely, the initial state of \( S(p) \) satisfies \( \varphi \) if and only if the initial state of \( S_{O(\varphi)}(p) \) satisfies \( \varphi \) and the two transition systems are called \( O(\varphi) \)-equivalent. The abstraction we define is formula-driven, but, since it is bound to the set of occurring actions of a formula and not to its structure, different formulae with the same set of occurring actions can be equivalently verified on the same abstraction of \( S(p) \).
The interesting thing is that $S_{O(\phi)}(p)$ may have a smaller state space than $S(p)$; in fact, the set of states reachable by $\rightarrow_S O(\phi)$ (the states of $S_{O(\phi)}(p)$) is smaller than the set of states reachable by the standard relation $\rightarrow_S$ (the states of $S(p)$), if $O(\phi) \subset A$.

For example, consider the simple process $p = b.c.a.nil$ and the formula $\psi_2 = \langle a \rangle \emptyset tt$ of Example 2. $S(p)$ has four states: $p$, $c.a.nil$, $a.nil$, $nil$. $S_{O(\psi_2)}(p)$, with $O(\psi_2) = \{ a \}$, has only two states: $p$, and $nil$, i.e., the state reached by performing a sequence of actions not in $O(\psi_2)$, $bc$, followed by an action in $O(\psi_2)$, $a$. For this simple process we obtain a state space reduction of 50%. Obviously, the size of the reduction is heavily influenced by the dimension, with respect to $A$, of the sets $K$ and $R$ in the formula: if the union of all sets $K$ and $R$ is equal to $A$, no reduction will be obtained.

It is worth noting that the actions relevant for checking a selective property cannot be so easily retrieved from the corresponding mu-calculus formula. For example, consider the formula $\psi_4 = \langle a \rangle \emptyset tt$, whose meaning is: “it is possible to perform the action $a$ without performing any action before”, and the corresponding mu-calculus formula $\psi'_4 = \langle a \rangle \emptyset tt$. The previous process $p = b.c.a.nil$ does not satisfy $\psi_4$ and this result is correctly obtained on the transition system $S_{O(\psi_4)}(p) = S(p)$, since $O(\psi_4) = \{ a, b, c \}$. In this case, our method does not obtain any reduction, but, if we (wrongly) deduce from $\psi'_4$ that only the action $a$ matters, and check $\psi'_4$ on the transition system $S_{\{a\}}(p)$, we obtain a wrong result. As a further example, consider the property $\psi_1 = [a][b] \emptyset ff$ of Example 2 and the transition system having initial state $p_2$ in Fig. 1. Since $O(\psi_1) = \{ a, b \}$ the property can be checked on $S_{O(\psi_1)}(p_2)$, see Fig. 2(i), and results true again. In the standard mu-calculus, $\psi_1$ becomes $\psi'_1 = \nu Z[a] \emptyset ff \land [c] Z$, then, if we take into account only the actions $\{ a, c \}$, we build $S_{\{a,c\}}(p_2)$, see Fig. 2(ii), which has the same number of states of $S_{O(\psi_1)}(p_2)$, but which obviously produces a wrong result since $\psi'_1$ evaluates to false.

Other approaches have been proposed to reduce the state explosion problem by verifying the formula $\varphi$ on the system $S_{abs}$ that is an abstraction of the system $S$, see for example the cone-of-influence reduction approach [9]. Nevertheless this abstraction is conservative: the fact that $\varphi$ is satisfied on $S_{abs}$ implies that it is satisfied on $S$. The approach based on the selective mu-calculus logic does not have this limitation.

---

1 $S_{\{a\}}(p)$ contains only the state $p$ and the state nil.

Table 3
Satisfaction of a formula by a process

| $p \not|= ff$ | $p |= tt$ |
|----------------|---------|
| $p |= \psi \land \psi$ iff $p |= \psi$ and $p |= \psi$ | $p |= \psi \lor \psi$ iff $p |= \psi$ or $p |= \psi$ |
| $p |= (K)R \psi$ iff $\forall \rho', \forall \alpha \in K, p \Rightarrow_{K \cup R} \rho'$ implies $p' |= \psi$ | $p |= (\exists \alpha) \psi$ iff $\exists \rho', \exists \alpha \in K, p \Rightarrow_{K \cup R} \rho'$ and $p' |= \psi$ |
| $p |= vZ.\psi$ iff $p |= vZ^n.\psi$ for all $n$ | $p |= \mu Z.\psi$ iff $p |= \mu Z^n.\psi$ for some $n$ |

- $vZn.\psi$ and $\mu Zn.\psi$ are defined as: $vZ0.\psi = \text{tt}$, $\mu Z0.\psi = \text{ff}$, $vZ^{n+1}.\psi = \psi[vZ^n.\psi/Z]$, $\mu Z^{n+1}.\psi = \psi[\mu Z^n.\psi/Z]$

- $\psi[\psi/Z]$ indicates the substitution of $\psi$ for every free occurrence of the variable $Z$ in $\psi$.

Example 3. Reconsider the tea dispenser, from Example 1, and the property

$\phi = [\text{tea}]_{\text{only_sugar, nothing}} (\text{lemon})_{\text{coin}} \text{tt}$: “whenever the user presses the tea button and the machine has displayed neither the message only_sugar nor the message nothing, he/she can press the lemon button, without inserting another coin”.

Since $O(\phi) = \{\text{tea, only_sugar, nothing, lemon, coin}\}$, the abstract transition system $\mathcal{S}_{O(\phi)}(U \parallel M)$ has 18 states and 37 transitions.

3. Compositional tableau

In this section a compositional tableau proof system is presented, for checking the property expressed by a closed formula of the selective mu-calculus on the FCCS parallel process $p \parallel q$.\footnote{For sequential processes, the tableau proof system has the structure, adapted to deal with the selective operators, of the analogous tableau in [16].} We remark that the aim of this work is to perform model checking
on the parallel composition of sequential processes, whose internal behavior has been successfully checked in advance, and whose composition should now be examined, for verifying properties bound to the process interaction. Thus, hereafter, we assume that the actions belonging to the sets $K$ occurring in the modal operators, $\langle K \rangle_R$ and $[K]_R$, are communication actions only. In the following, we shall show in detail the behavior of the method when the parallel composition involves only two processes, while the extensions to handle more processes will be suggested by some hints in the proper points.

The rules of the system operate on sequents, called $P$-sequent.

**Definition 2 ($P$-sequent).** Let $\varphi$ be a selective mu-calculus formula. A $P$-sequent is an expression of the form

$$S \vdash_{\Delta}^{M_1, M_2} \varphi$$

where:

- $M_1$ and $M_2$ are two transition systems for sequential processes.
- $\Delta$ is a definition list, i.e., a sequence of declarations $U_1 = \varphi_1, \ldots, U_n = \varphi_n$, such that $U_i \neq U_j$, if $i \neq j$ and such that each constant occurring in $\varphi_i$ is one of $U_1, \ldots, U_{i-1}$.
  The operation $\Delta \cdot U = \varphi$ is defined to append the declaration $U = \varphi$ to the definition list $\Delta$.
- $S = \{(S_{11}, S_{21}), \ldots, (S_{1n}, S_{2n})\}$ is called $P$-set and is a set of pairs where each $S_{1i}$ and $S_{2i}$ is a subset of the states of $M_1$ and $M_2$, respectively.\(^3\)

Each rule is of the form:

$$S \vdash_{\Delta}^{M_1, M_2} \varphi$$

$$S \vdash_{\Delta_1}^{M_1, M_2} \varphi_1 \quad \cdots \quad S \vdash_{\Delta_n}^{M_1, M_2} \varphi_n$$

where $n > 0$. The premise sequent is the goal to be achieved, the consequents are the sub-goals which are determined by the structure of the formula, by $\Delta$ and by the moves possible in the two models, $M_1$ and $M_2$, starting from the states present in $S$. Often, in the sequel, the indexes $M_1$ and $M_2$ are dropped from the $P$-sequents.

Suppose we would like to check whether $p \parallel q$ has the property $\varphi$: the initial goal is represented by a $P$-sequent of the form: $\{(s_p, s_q)\} \vdash_{S(p), S(q)} \varphi$, where $s_p$ and $s_q$ are the initial states of $S(p)$ and $S(q)$, respectively. The definition list at this point is empty and then omitted. Thus each sequential process in a parallel composition is represented, at each verification step, by a (sub-)set of the states of its transition system. Each pair of the $P$-set relates states of one process to states of the other one: the relation is established by means of two operators which are applied to a $P$-set to produce a new $P$-set. The first operator is presented in the following definition.

---

\(^3\) When more than two processes are composed, $S_{1i}$ and $S_{2i}$ can be again set of pairs, and so on.
Definition 3 ($\mathcal{F}_j$). Let $C, R \subseteq A$, $K \subseteq C$, $\alpha \in K$, and let $S$ be a P-set.

$$\mathcal{F}_j(S, K, C, R) = \begin{cases} (S_1, S_2) \in S, & p \in S_1, \ q \in S_2, \\ \exists p', q'. & p \xrightarrow{\delta \alpha} C \cup R p', \ q \xrightarrow{\delta \alpha} C \cup R q', \\ \Pi_{R \cup K}(\delta) = \epsilon \\ S_1' = \{ p' | p \xrightarrow{\delta \alpha} C \cup R p' \}; \ and \\ S_2' = \{ q' | q \xrightarrow{\delta \alpha} C \cup R q' \} \end{cases}.$$  

Intuitively, if $p$ and $q$ are two states in $S$ and, after performing the same sequence of actions $\delta \alpha$, where $\delta$ is composed of communication actions not belonging to $R \cup K$, can evolve to the states $p'$ and $q'$, respectively, $\mathcal{F}_j$ returns the pair $(p', q')$. The actual sequence of actions performed by $p$ and $q$ may be $\delta$ interleaved by non-interesting local actions; thus, $\xrightarrow{C \cup R}$ while finding an action in $K$, discards some states of the sequential processes, i.e., the states reached by arcs labeled with actions not in $C \cup R$. The condition $\Pi_{R \cup K}(\delta) = \epsilon$ guarantees that $\alpha$ is the first encountered action of $K$ and that it is preceded by no action in $R$.

The following example shows in more details the behaviour of $\mathcal{F}_j$.

Example 4. Consider the P-set $\{(p, q)\}$, where $p$ and $q$ are the following two FCCS processes:

$$p = e.c.a.p_1 + c.f.a.b.p_2$$
$$q = c.a.q_1 + h.c.a.q_2 + c.a.b.q_3$$

Given $C = \{a, b, c\}$, $R = \emptyset$, $K = \{a\}$.

$$\mathcal{F}_j(\{(p, q)\}, \{a\}, \emptyset) = \{((p_1, b.p_2) \{q_1, q_2, b.q_3\})\}.$$  

On the other hand, whenever $p \parallel q$ performs the action $a$ it reaches (by using $\xrightarrow{c0}_{[a, b, c]}$) one of the following states:

$$p_1 \parallel q_1 \ \ p_1 \parallel q_2 \ \ p_1 \parallel b.q_3 \ \ b.p_2 \parallel q_1 \ \ b.p_2 \parallel q_2 \ \ b.p_2 \parallel b.q_3.$$  

Thus $\mathcal{F}_j$ stores only some of the states of the sequential processes and does not combine them, so reducing state space explosion.

The other operator is the following.

Definition 4 ($\mathcal{F}_0$). Let $C, R \subseteq A$, $K \subseteq C$, $\alpha \in K$, and let $S$ be a P-set.

$$\mathcal{F}_0(S, K, C, R) =$$

---

It is worth noting that, when more than two processes are composed in parallel, for example $p = (p_1 \parallel p_2)$, it is possible that the move $p \xrightarrow{L(p_2)} p'$ be obtained by moving only $p_1 \xrightarrow{L(p_2)} p'_1$, since $\alpha \not\in L(p_2)$, thus also this kind of move must be included in the definition of $\mathcal{F}_j$. 
∃(S₁, S₂) ∈ S, ∃p ∈ S₁, ∃q ∈ S₂  
\text{s.t. } \mathcal{F}_c(p, q, K, C, R) = \emptyset

\begin{align*}
\mathcal{F}_c(p, q, K, C, R) =
\begin{cases}
\emptyset & \text{if } p \xrightarrow{\delta a} C ∪ R p' \\
\{(p'), (q')\} & \text{if } q \xrightarrow{\delta a} C ∪ R q' \text{ and } \Pi_{R∪K}(\delta) = \varepsilon \\
\emptyset & \text{otherwise.}
\end{cases}
\end{align*}

If both \( p \xrightarrow{\delta a} C ∪ R p' \) and \( q \xrightarrow{\delta a} C ∪ R q' \) are moves of \( p \), with a corresponding move of \( q, q \xrightarrow{\delta a} C ∪ R q' \), the result of the application of \( \mathcal{F}_0 \) is either \((p', q')\) or \((p'', q')\), but not \((p', p''), q'\). In fact, the nondeterministic choice operator \( \mathcal{F}_c \) performs a unique choice among the equal moves of each sub-process. Moreover, given a P-set \( S = \mathcal{F}_0 \) produces a new not empty P-set only if, for each pair in \( S \), for example \( \{(p_1, p_2), (q_1, q_2)\} \), at least a corresponding move exists for each \( p_i, i, j \in 1, 2 \). These two characteristics of \( \mathcal{F}_0 \) are further explained by the following example.

\textbf{Example 5.} Reconsider the two processes of \textbf{Example 4}. A possible result of \( \mathcal{F}_0(\{(p, q)\}, \{a\}, C, \emptyset) \) is \( \{(p_1, q_1)\} \), in fact, only one move computed by \( \xrightarrow{cd} \) for \( p \) and for \( q \) is stored. Other results are, for example, \( \{(p_1, q_2)\}, \{(b, p_2, q_1)\} \).

Now, reconsider the result of \( \mathcal{F}_1(\{(p, q)\}, \{a\}, C, \emptyset) \) in \textbf{Example 4} that is \( S = \{(p_1, b.p_2) [q_1, q_2, b.q_3]\} \) and apply \( \mathcal{F}_0(S, \{b\}, C, \emptyset) \); moreover, suppose that \( p_1, q_1, q_2 \) contain no action \( b \).

\( \mathcal{F}_0(S, \{b\}, C, \emptyset) = \emptyset \) (even if an action \( b \) can be performed), since there exists at least one pair of sub-processes, for example, \( p_1 \) and \( q_1 \), which, when composed in parallel, are not able to perform \( b \).

Now apply \( \mathcal{F}_0(\{(p, q)\}, \{a\}, C, \emptyset) \): a possible result is \( \{(b, p_2, b.q_3)\} \). \( \mathcal{F}_0(\{(b, p_2, b.q_3)\}, \{b\}, C, \emptyset) \) produces \( \{(p_2, q_3)\} \).

Note that \( \mathcal{F}_0 \) is used to deal with the \( \langle K \rangle_R \) modal operator, while \( \mathcal{F}_1 \) is used to deal with the \( \mathcal{K}_R \) one.

Finally, a \textit{cleaning function} \( Cl \) is defined, which examines a P-set and reduces the possible replication of states in the pairs by compacting in a single one the pairs with equal first component and, then, the pairs with equal second component. In fact, once a state, for example, \( p' \), has been reached by moving \( p \), it does not matter if it has been reached after different communication patterns in different global moves with \( q \); in the transition system of \( p \parallel q, p' \) is combined with all the states which \( q \) reaches after any according move.

\textbf{Definition 5 (Cl)}. Consider a P-set \( S. Cl(S) = S', \) where \( S' \) is the new P-set so obtained:

\( S' \leftarrow S; \)

while \( \exists(A, B), (C, D) \in S' \) such that \( A = C \) or \( B = D \) do

begin
if \( \exists(A, B), (A, C) \in S' \) then \( S' \leftarrow S' - \{(A, B), (A, C)\} \cup \{(A, B \cup C)\} \)
else if \( \exists(A, B), (C, B) \in S' \) then \( S' \leftarrow S' - \{(A, B), (C, B)\} \cup \{(A \cup C, B)\} \)
end
Suppose \( S = \{(p, q_1), (p, q_2), (p, q_3), (r, s)\} \). It holds that:
\[
\text{Cl}(S) = S' = \{(p, \{q_1, q_2, q_3\}), (r, s)\}.
\]

Now, we give a complete example of the use of our method.

**Example 6.** Consider the following two FCCS processes, and their parallel composition with \( C = \{a, b, c, d\} \):

\[
p = c.a \cdot e.b.nil + d.l.a \cdot e.nil
\]

\[
q = c.a \cdot b.nil + h.c.a \cdot f.h.nil + d.a \cdot b.nil + d.a \cdot f.h.nil + c.a \cdot g.nil + d.a \cdot g.nil.
\]

We would like to check the following two properties:

1. \( [a]_\emptyset \{b\}_\emptyset \vdashtt \) “After each action \( a \), it is possible to perform the action \( b \) not preceded by the action \( e \)”,
2. \( (a)_\emptyset \{b\}_\emptyset \vdashtt \) “It is possible to perform the action \( a \) followed by the action \( b \)”.

Consider the first property:

\[
\mathcal{F}_1(\{(p, q), \{a\}, C, \emptyset\}) = S_1 = \{(p_1, \{q_1, q_2, q_3\}), (p_2, \{q_1, q_2, q_3\})\},
\]

and

\[
\text{Cl}(S_1) = S_2 = \{(p_1, p_2), \{q_1, q_2, q_3\}\}.
\]

Then, \( \mathcal{F}_2(\{S_1, \{b\}, C, \{e\}\}) = \emptyset \), since \( p_1 \) has no move of the form \( p_1 \xrightarrow{bb} \cup_{C\{e\}} p' \) such that \( \Pi_{C\{e\}|\{b\}}(\delta) \neq e \). This means that \( p \parallel q \) does not satisfy \( [a]_\emptyset \{b\}_\emptyset \vdashtt \).

Consider now the second property:

\[
\mathcal{F}_2(\{(p, q), \{a\}, C, \emptyset\}) = S_3 = \{(p_1, q_1)\},
\]

and

\[
\mathcal{F}_2(\{S_3, \{b\}, C, \emptyset\}) = \{\{nil, nil\}\}.
\]

This means that \( p \parallel q \) satisfies \( (a)_\emptyset \{b\}_\emptyset \vdashtt \).

The rules of the tableau system are shown in Table 4. We assume that \( \sigma \) ranges over \( \{\mu, \nu\} \).

It is important to point out that the rule \( () \) corresponds to a set of rules, one for each of the possible results of \( \mathcal{F}_2(\{S, K, C, R\}) \).

A tableau for \( S \vdash \mathcal{M}_1, \mathcal{M}_2, \varphi \) is a maximal proof tree whose root is labeled with that \( P \)-sequent. The \( P \)-sequents labeling the immediate successors of a node labeled \( S \vdash \varphi \) are determined by an application of one of the rules in Table 4, depending on the structure of \( \varphi \). Maximal means that no rule applies to a \( P \)-sequent labeling a leaf of a tableau. We assume that the rules in Table 4 only apply to nodes of a proof tree that are not terminal. A node \( n \) labeled with the \( P \)-sequent \( S \vdash \varphi \) is terminal if one of the following conditions holds:
Successful terminal

1. \( \varphi = \text{tt} \).
2. \( \varphi = [K]R \varphi' \) and 
   \( \mathcal{F}_{[1]}(S, K, C, R) = \emptyset \).
3. \( \varphi = U, U = vZ.\varphi' \) and
   there is a node above \( n \) labeled \( S \vdash \varphi \).

Thus the goal \( \{ (s_p, s_q) \} \vdash_{S(p) \cdot S(q)} \varphi \), where \( s_p \) and \( s_q \) are the initial states of the transition systems \( S(p) \) and \( S(q) \) with \( p \) and \( q \) sequential processes, is obtained by building a successful tableau, i.e., a finite proof tree whose root is labeled with the \( P \)-sequent representing the goal and where all leaves are successful (i.e., they obey one of the conditions 1, 2, 3 above).

Two important theorems follow for finite FCCS processes. Theorem 1 amounts to decidability, while Theorem 2 states that our tableau technique is both sound and complete.

**Theorem 1.** Let \( p \parallel q \) be a FCCS term: every tableau for \( \{ (s_p, s_q) \} \vdash \varphi \) is finite.

**Proof.** Ad absurdum.\(^5\) □

**Theorem 2.** Let \( p \parallel q \) be an FCCS process and \( \varphi \) a selective mu-calculus formula.

\( \{ (s_p, s_q) \} \vdash \varphi \) has a successful tableau \( \Leftrightarrow p \parallel q \models \varphi \).

To prove the theorem we need first the following two technical lemmas:

**Lemma 1.** Let \( S \) be a \( P \)-set. It holds that:

1. \( \mathcal{F}_{[1]}(S, K, R, C) = \mathcal{F}_{[1]}(\mathcal{C}(S), K, R, C) \); and
2. \( \mathcal{F}_{0}(S, K, R, C) = \mathcal{F}_{0}(\mathcal{C}(S), K, R, C). \)

\(^5\) Note that, as stated in Section 2.1, we consider only finite FCCS processes.
Proof. Let $\text{number}(S)$ be the number of pairs $(A, B)$, $(C, D)$ in $S$ such that $A = C$ or $B = D$. The proof proceeds by induction on $n = \text{number}(S)$. We prove only the first point; the second one can be proved similarly.

Base step. $n = 0$: straightforward, since $S = \text{Cl}(S)$, by Definition 5.

Inductive step. The lemma holds for $n$: we prove the lemma for $n + 1$. Suppose $S' = S \cup \{(A, B)\} \cup \{(A, C)\}$ with $\text{number}(S') = n$ and $\text{Cl}(S') = \text{Cl}(S) \cup \{(A, B \cup C)\}$.

By Definition 3, $(S'_1, S'_2) \in \mathcal{F}_1(\hat{S}, K, R, C)$, if $\exists p', q'$ and $\alpha \in K$ such that $S'_1 = \{p' \mid p \xrightarrow{\delta \alpha} C \cup R p'\}$ and $S'_2 = \{q' \mid q \xrightarrow{\delta \alpha} C \cup R q'\}$, where $(\hat{S}_1, \hat{S}_2) \in \hat{S}$, $p \in \hat{S}_1$, $q \in \hat{S}_2$ and $I_{R \cup K}(\delta) = \emptyset$.

If $(S_1, S_2) \in S$, the thesis follows by the inductive hypothesis; otherwise, assume $(\hat{S}_1, \hat{S}_2) = (A, B)$ (similarly, for $(A, C)$), if $p \in \hat{S}_1$ and $q \in \hat{S}_2$, then for $\text{Cl}(S')$ it is $p \in A$ and $q \in B \cup C$. Vice versa, assume $(\hat{S}_1, \hat{S}_2) = (A, B \cup C)$, if $p \in \hat{S}_1$ and $q \in \hat{S}_2$, in $S'$ it is also $p \in A$ and either $q \in B$ or $q \in C$. □

Lemma 2. Let $S$ be a P-set and $\varphi$ a selective mu-calculus formula.

$$S \vdash \varphi \Leftrightarrow \text{Cl}(S) \vdash \varphi.$$  

Proof Sketch. We prove only the direction $S \vdash \varphi \Rightarrow \text{Cl}(S) \vdash \varphi$. The interesting cases involve the inference rules for the selective modal operators, so we consider only $(K)_R \varphi'$ and $(K)_R \varphi'$ formulae. The proof proceeds by induction on the structure of the formula.

$(K)_R \varphi'$. By definition of $(\cdot)$ rule of Table 4, we have that

$$S \vdash (K)_R \varphi' \text{ if } \text{Cl}(\mathcal{F}_0(S, K, C, R)) \vdash \varphi'.$$

The thesis follows by inductive hypothesis and by point 2 of Lemma 1.

$(K)_R \varphi'$. Can be proved similarly, using point 1 of Lemma 1. □

Theorem 2 is now an immediate consequence of the successive lemma:

Lemma 3. $S \vdash \varphi \Leftrightarrow \forall(S_1, S_2) \in S, \forall p \in S_1, \forall q \in S_2, \text{ it holds that } p \parallel q \models \varphi.$

Proof Sketch. The interesting cases involve the inference rules for the selective modal operators, so we consider only $(K)_R \varphi'$ and $(K)_R \varphi'$ formulae.

$(\Leftarrow)$ The proof proceeds by induction on the structure of the formula.

$$(K)_R \varphi'$$

$$\forall(S_1, S_2) \in S, \forall p \in S_1, \forall q \in S_2 \text{ it holds that } p \parallel_C q \models (K)_R \varphi'$$

{ Definition of satisfaction (see Table 3 ) }

$$\forall(S_1, S_2) \in S, \forall p \in S_1, \forall q \in S_2 \text{ it holds that } \forall r \models (K)_R \varphi'$$

{ Operational semantics, Definition 1 and $K \subseteq C$ }

$^6$ The case $S' = S \cup \{A, B\} \cup \{B, C\}$ is similar.
∀(S₁, S₂) ∈ S, ∀p ∈ S₁, ∀q ∈ S₂ it holds that ∀p', q'. ∀α ∈ K. p \xrightarrow{δα} C \cup R p', q \xrightarrow{δα} C \cup R q' and \Pi_{K \cup R}(δ) = ε, implies p' \parallel_C q' |= ϕ' { Inductive Hypothesis and definition of \mathcal{F}_\{\} }

\mathcal{F}_\{\}(S, K, R, C) \vdash ϕ' { Lemma 2 }

\mathcal{C}(\mathcal{F}_\{\}(S, K, R, C)) \vdash ϕ' { Rule \{\} }

S \vdash [K]_R ϕ'.

\[K\]_R ϕ':
∀(S₁, S₂) ∈ S, ∀p ∈ S₁, ∀q ∈ S₂ it holds that p \parallel_C q |= (K)_R ϕ' { Definition of satisfaction (see Table 3) }

∀(S₁, S₂) ∈ S, ∀p ∈ S₁, ∀q ∈ S₂ it holds that \exists r. \exists α ∈ K. p \parallel_C q \xrightarrow{α} K \cup R r and r |= ϕ' { Operational semantics Definition 1 and K ⊆ C }

∀(S₁, S₂) ∈ S, ∀p ∈ S₁, ∀q ∈ S₂ it holds that \exists p', q'. \exists α ∈ K. p \xrightarrow{δα} C \cup R p', q \xrightarrow{δα} C \cup R q', \Pi_{K \cup R}(δ) = ε and p' \parallel_C q' |= ϕ' { Inductive Hypothesis and definition of \mathcal{F}_\{\} }

\mathcal{F}_\{\}(S, K, R, C) \vdash ϕ' { Lemma 2 }

\mathcal{C}(\mathcal{F}_\{\}(S, K, R, C)) \vdash ϕ' { Rule \{\} }

S \vdash (K)_R ϕ'.

⇒ The proof proceeds by induction on the structure of the formula.

[ K ]_R ϕ':
S \vdash [K]_R ϕ' { Rule \{\} of Table 4 }

\mathcal{C}(\mathcal{F}_\{\}(S, K, C, R)) \vdash Δ ϕ' { Lemma 2 }

\mathcal{F}_\{\}(S, K, C, R) \vdash Δ ϕ'.

The thesis follows by inductive hypothesis and by definition of satisfaction (see Table 3).

( K )_R ϕ': similarly. □

**Example 7.** The tea dispenser, from Example 1, should have the property

ϕ : [tea]_{only_sugar,nothing} (lemon)_{coin} tt

of Example 3. Fig. 3 shows the standard transition systems of \(U\) and \(M\), while Fig. 4 shows the interaction structure of the tea dispenser, which is obtained as the parallel composition of the processes \(U\) and \(M\) over the set of communication actions \(C\).

Below is a successful tableau for \(\{ (U, M) \} \vdash ϕ\). By Theorem 2, we can deduce that \(U \parallel M\) has the property \(ϕ\).
Fig. 3. The standard transition systems of $U$ and $M$. 
\[
\{U, M\} \vdash \text{tea}\{\text{only} \_ \text{sugar}, \text{nothing}\}\text{lemon}\{\text{coin}\}\text{tt}
\]

\[
S_1 \vdash \text{lemon}\{\text{coin}\}\text{tt}
\]

\[
S_2 \vdash \text{tt}
\]

where
\[
C = \{\text{coin, tea, sugar, lemon, help, go}\}
\]
\[
S_1 = \{\{U_1, U_2\}, \{M_1, M_2\}\}
\]
\[
S_2 = \{\{U_3, M_5\}\}.
\]

\(U_1, U_2\) (resp. \(M_1, M_2\)) are the states reached by the process \(U\) (resp. \(M\)) after performing \text{tea} not preceded neither by \text{only} \_ \text{sugar} nor by \text{nothing} (see Fig. 3). Note that, with respect to the 43 states of the standard transition system for \(U \parallel M\), and the 18 states of the abstract transition system of Example 3, the states which are considered during the proof are 8.

4. Discussion

In this section we discuss the results obtained by means of our method in reducing the state space during the verification of selective mu-calculus formulae. The presented method combines three well-known approaches: local model checking, compositionality and abstraction techniques; in this way, the advantages of each approach are joined in only one methodology.

Local model checking \([10,16]\) is a known verification technique for the temporal logic mu-calculus. First we discuss how the selective mu-calculus logic improves this method from the point of view of the state space reduction.
Suppose we have to check whether \( p = b.c.a.h.nil \parallel d.h.nil \), with \( C = \{ h \} \), satisfies the selective formula

\[
\varphi = [h]_a \quad \text{ff}
\]

that means: “it is not possible to perform the action \( h \) without having previously performed the action \( a \)”. We recall that, as shown in Section 2.2, this formula corresponds to the following \( \mu \)-calculus one:

\[
\varphi' = vZ.[h] \quad \text{ff} \land [A - \{a, h\}]Z.
\]

The tableau obtained following the methods proposed in [10,16] explores almost all states of \( S(p) \) to determine that \( p \) satisfies \( \varphi' \). If we apply the rules in Table 4, it is easy to see that only the initial state has to be stored to establish the same result for \( \varphi \). In this case, as in general, the degree of reduction mainly depends on the number of occurring actions of a formula with respect to the number of actions in the set \( A \). It is worth noting that the states we do not store in the construction of the proof must be in any case traversed during the computation of the single move; nevertheless, after a move is found, the states stored during its computation can be discarded and only the final state is kept.

Now, assume we have to check whether \( b.c.p_1 + b.c.p_2 + b.c.p_3 \parallel b.c.q_1 + b.c.q_2 + b.c.q_3 \) satisfies the following selective formula, with \( O(\varphi) = A \):

\[
\psi = [A - \{c\}]_A (c).\quad \text{tt}.
\]

When no action can be ignored, no abstraction can be made. The \( \mu \)-calculus formula corresponding to \( \psi \) is \( \psi' = [A - \{c\}] (c).\quad \text{tt} \).

Using again the methods proposed in [16,10], all the following states, besides the initial one, are stored:

\[
\begin{align*}
&c.p_1 \parallel c.q_1, \quad c.p_1 \parallel c.q_2, \quad c.p_1 \parallel c.q_3, \quad c.p_1 \parallel c.q_3, \quad c.p_2 \parallel c.q_2, \quad c.p_2 \parallel c.q_3, \\
&c.p_3 \parallel c.q_1, \quad c.p_3 \parallel c.q_2, \quad c.p_3 \parallel c.q_3, \quad p_1 \parallel q_1
\end{align*}
\]

while our method stores only the following two \( P \)-sets, besides the initial one:

\[
\{(c.p_1, c.p_2, c.p_3), \{c.q_1, c.q_2, c.q_3\}\} \quad \text{and} \quad \{(p_1, q_1)\}.
\]

So it is worth noting that the proposed method, exploiting both abstraction and compositionality, is not bound to the selective \( \mu \)-calculus, but can be applied also to other logics. Moreover, using abstraction techniques alone on the whole parallel process, we obtain a lesser reduction than using the full method on the separate sequential processes; see for this fact, Examples 3 and 7, where the same formula is first checked on the abstract transition system for the parallel process of Example 1, and then by our method. From another point of view, we can quantify the degree of improvement we obtain, with respect to using a theoretical compositionality technique alone, by considering the abstract transition systems, \( S_\rho(U) \) and \( S_\rho(M) \), for the sequential processes in the Example 1, built by the transition relation \( \approx_\rho \), where \( \rho = O(\varphi) \cup C \). During the proof for the goal \( \{U, M\} \vdash \varphi \), the dimension of the state space to be explored is less than or equal to
\[ m' + n', \text{ where } m' \text{ and } n' \text{ are the number of states of } S_\rho(U) \text{ and } S_\rho(M), \text{ respectively.} \]

Such values result equal to \( m \) and \( n \), respectively the states of \( S(U) \) and \( S(M) \), when no abstraction can be made.

As a final consideration, note that in the above examples each state of the transition systems of the sequential processes is stored once. Nevertheless, there are cases in which this is not true. For instance, suppose that we want to check whether the following process, with all actions in \( C \):

\[ c.a.p + d.a.q + f.a.t + b.a.k \parallel d.a.r + c.a.r + c.a.s + f.a.s + b.a.s + d.a.s \]

satisfies \([a]\psi\phi\): “after an action \( a \) has occurred, preceded by any action, \( \psi \) holds”.

In this case, with our methodology, we store the following \( P \)-set:

\[ \{(t, k), \{s\}\}, \{(p, q), \{r, s\}\} \]

The state \( s \) is stored twice, since different communication paths exist in \( Y \) leading to \( s \): but different communication paths exist due to the existence of intermediate different states along these paths, which are not stored in any \( P \)-set. Thus, however, we store a number of states which is less than (or equal to) the sum between the number of states of \( S(X) \) and of \( S(Y) \).

5. Related works and conclusions

Model checking algorithms for temporal logic such as the alternation-free fragment of \( \mu \)-calculus are well known to run in time \( O(n \cdot m) \), where \( n \) is the size of the transition system and \( m \) is the length of the formula to be checked. Since the state space of the transition system, due to the possible interleaving of the concurrent actions, can grow exponentially, it is relevant for the efficiency of the approach, to avoid building the entire state space. Much of the work on reducing the complexity of automatic verification can be grouped into two classes. In the first class are the methods which try to build only the needed part of the global state graph. This purpose can be achieved by using local model checking algorithms [10,16], which for \( \mu \)-calculus use tableau-based proofs to deduce that the system (actually the initial state of the transition system modeling the system) satisfies a given formula. While, for some formulae, only a small portion of the state space may have to be examined, it often occurs that the entire state space is generated, when checking that a property holds globally. The same drawback exists also using abstraction techniques such as that based on the selective \( \mu \)-calculus, since formulae such as the one stating deadlock freedom must make visible all actions occurring in the system, and then no abstraction can be made on the state space. It is possible to verify that this problem occurs also in the majority of the methods trying to reduce the global state space or to building only the needed part.

The methods of the other class rely on compositionality: properties of the global system are deduced from properties of its sub-components. In this case the global state space

\[ 7 \text{ It holds that } m = 11, n = 20, m' = 9 \text{ and } n' = 15. \]
is not built in any case: the key point is to identify and use the properties of the sub-
systems. Works exist (see [11]) in which it is proved that, for a given property $\varphi$ that
a concurrent process must satisfy, there exist properties $\varphi_1$, $\varphi_2$ that the processes in
the composition must satisfy. Nevertheless, no method is given to deduce such properties.
Other works (see [1]), given the property that the concurrent process must satisfy, propose
a method to deduce the property which one of the composed process must satisfy; the
drawback is that it results in an explosion of the length of the original formula because
the parallel composition is eliminated by encoding one of the component of the parallel
composition into the formula: so an effective reduction of the complexity of the model
checking procedure is not achieved. Finally, for example, in [6], the properties which the
sub-components satisfy are known, while the property which must hold for their parallel
composition is inferred by theorem proving. This approach has the drawback that theorem
proving is not an automatic procedure.

The approach we present exploits the advantages of both classes using compositionality
to traverse, in the worst case, only the sum of the states of the transition systems of the
sequential components of a parallel composition, and relying on local model checking to
traverse, in the best case, only a part of those states. The use of selective $\mu$-calculus further
reduces the number of states to be considered, since they are only those ones in which a
visible action leads; in the worst case the bound $m+n$ is not exceeded, where $m$ and $n$ are
the size of the transition systems of the sequential components of a parallel composition.
However, it is worth noting that the method is not bound to this logic, but can be usefully
employed also in model checking $\mu$-calculus, simply considering visible all actions. In
conclusion, it is possible to say that the use of various techniques in the same method
permits us to improve both the best case (when abstraction and local model checking allow
the exploration of a part of the state space of a system) and the worst case (when the
formula to be checked requires the exploration of all the states, the compositionality of the
proof keeps a linear bound for the state space).

References

[1] H.R. Andersen, C. Stirling, G. Winskel, A compositional proof system for modal $\mu$-calculus,
proving properties on reduced transition systems, in: Proceedings of FORTE X/PSTV XVII’97, Chapman
Workshop on Computer Aided Verification, CAV’92, Lecture Notes in Computer Science, vol. 663, 1992,
1997.
in: Proceedings of the 18th International Colloquium on Automata, Languages and Programming, Lecture