32
Selective mu-calculus: New Modal Operators for Proving Properties on Reduced Transition Systems

Roberto Barbuti
Dipartimento di Informatica. Università di Pisa, 56125 Pisa, Italy.
e-mail: barbuti@di.unipi.it

Nicoletta De Francesco, Antonella Santone, Gigliola Vaglini
Dipartimento di Ingegneria dell'Informazione. Università di Pisa, 56126 Pisa, Italy.
e-mail: {nico,santone,gigliola}@iet.unipi.it

Abstract

In model checking for temporal logic, the correctness of a (concurrent) system with respect to a desired behavior is verified by checking whether a structure that models the system satisfies a formula describing the behaviour. Most existing verification techniques, and in particular those defined for concurrent calculi like as CCS, are based on a representation of the concurrent system by means of a labelled transition system. In this approach to verification, state explosion is one of the most serious problems. In this paper we present a new temporal logic, the *selective mu-calculus*, with the property that only the actions occurring in a formula are relevant to check the formula itself. We prove that the selective mu-calculus is as powerful as the mu-calculus. We define the notion of ρ-bisimulation between transition systems: given a set of actions ρ, a transition system ρ-bisimulates another one if they have the same behaviour with respect to the actions in ρ. We prove that, if two transition systems are ρ-equivalent, they preserve all the selective mu-calculus formulae with occurring actions in ρ. Consequently, a formula with occurring actions ρ can be more efficiently checked on a transition system ρ-equivalent to the standard one, but smaller than it.

Keywords

Mu-Calculus, State Explosion, Abstraction, CCS

Formal Description Techniques and Protocol Specification, Testing and Verification
T. Mizuno, N. Shiratori, T. Higashino & A. Togashi (Eds.)

1 INTRODUCTION

In model checking for temporal logic, the correctness of a (concurrent) system with respect to a desired behavior is verified by checking whether a structure that models the system satisfies a formula describing the behaviour. Most existing verification techniques, and in particular those defined for concurrent calculi like as CCS [23], are based on a representation of the concurrent system by means of a labelled transition system [8, 12]. In this approach to verification, state explosion is one of the most serious problems: systems are often described by transition systems with a prohibitive number of states. On the other hand, in several cases, it is sufficient to verify a property on a reduced transition system containing only the "parts" which "influence the property". Thus a solution to state explosion is the definition of suitable abstraction criteria by means of which a reduced transition system can be obtained, which abstracts from the parts not concerned with the property to be verified. The works [3, 22, 24, 25, 26, 28, 29] deal with abstractions of transition systems preserving only properties expressible by sub-languages of a general temporal logic language, for example avoiding the use of some operators. The works [1] and [10] present methods for constructing reduced transition systems, where the reduction is based on a temporal logic formula: the reduced system preserves the truth value of the formula. However, [10] refers only to formulae written in a subset of CTL logic, while the method in [1] can be applied only to systems obtained as the composition (product) of smaller ones. In both cases, the reduced transition system is obtained by means of a non-trivial algorithm. Other methodologies exist in which abstraction criteria are issued by the user of the verification environment [6, 7, 12, 13]; although useful in practice, this approach cannot be automated.

Since our aim is to obtain reductions in an automatic way from a formula expressing a temporal property, we consider two main aspects: the first one is the definition of a formalism suitable to express such properties; the second one is the method for extracting, from the definition of a property, the information sufficient to characterize the reduced transition systems. A suitable formalism to express temporal properties could be the modal mu-calculus extended with fixpoint formulae [27]. However, this formalism, although very powerful, cannot be used for easily deducing, from a formula, the reduction which can be performed on the standard transition system to obtain a smaller one on which the formula can be equivalently checked (this point will be discussed extensively in the following).

In order to cope with this problem, we define a different calculus, called *selective mu-calculus*, obtained by replacing the modal operators of the mu-calculus by new "selective modal operators". This new calculus has the same power of the original one: the mu-calculus can be expressed by means of the selective mu-calculus, and viceversa. In addition, each formula written using the selective operators allows us to immediately point out the parts of the transition

system that can be disregarded in checking the formula. To formalize this fact, we define the notion of ρ-equivalence between transition systems: given a set of actions ρ, two transition systems T_1 and T_2 are ρ-equivalent iff they present the same behaviour with respect to the actions in ρ. We prove that, if two transition systems are ρ-equivalent, they preserve all the formulae such that the set of actions occurring inside the modal operators of the formulae is a subset of ρ. Thus, to prove a formula, with occurring actions ρ, we check it on a transition system which is ρ-equivalent to the standard one, but which contains only the actions in ρ.

We would like to remark the elegance and the simplicity of our approach: the selective mu-calculus is very easy to understand and to use, being a slight modification of standard mu-calculus. Nevertheless, differently from mu-calculus, its formulae can be proved on reduced transition systems, the structure of which is suggested by the formulae themselves.

After the preliminaries in Section 2 and an informal overview of the approach in Section 3, we define the selective mu-calculus in Section 4. Experimental results are given in Section 5 and Section 6 concludes the work. The proofs of the theorems are only sketched. The complete proofs can be found in [2].

2 PRELIMINARIES

2.1 The Calculus of Communicating Systems

Let us now quickly recall the main concepts about the Calculus of Communicating Systems (CCS) [23]. The syntax of *process expressions* (*processes* for short) is the following:

$$P ::= nil | X | \alpha.P | P + P |\ P|P\ |P\backslash L | P[f]$$

where α ranges over a finite set of actions $\mathcal{A} = \{\tau, a, \overline{a}, b, \overline{b}, ...\}$. The action $\tau \in \mathcal{A}$ is called the *internal action.* The set of *visible actions*, $\mathcal{L}$, ranged over by $l, l' ...$, is defined as $\mathcal{A} - \{\tau\}$. Each action $l \in \mathcal{L}$ (resp. $\overline{l} \in \mathcal{L}$) has a *complementary action* $\overline{l}$ (resp. l). X ranges over a set of *constant* names: each constant X is defined by a constant definition $X \stackrel{def}{=} P$. We denote the set of process expressions by $\mathcal{E}$.

An *operational semantics* is a transition relation $\longrightarrow_O \subseteq \mathcal{E} \times \mathcal{A} \times \mathcal{E}$, where $\mathcal{E}$ is the set of all the processes. If $(P, \alpha, Q) \in \longrightarrow_O$, we write $P \stackrel{\alpha}{\longrightarrow}_O Q$. The standard semantics of CCS as defined in [23], will be denoted by $\longrightarrow_S$.

Given an operational semantics $\longrightarrow_O$, if $\delta \in \mathcal{A}^*$ and $\delta = \alpha_1 \ldots \alpha_n, n \geq 1$, we write $P \stackrel{\delta}{\longrightarrow}_O Q$ to mean $P \stackrel{\alpha_1}{\longrightarrow}_O \cdots \stackrel{\alpha_n}{\longrightarrow}_O Q$. For the empty sequence of actions $\lambda \in \mathcal{A}^*$ we have $P \stackrel{\lambda}{\longrightarrow}_O P$. With $\mathcal{D}_O(P) = \{Q | P \stackrel{\delta}{\longrightarrow}_O Q\}$ we denote the set of the *derivatives* of P by $\longrightarrow_O$.

A term P is *image finite* by an operational semantics $\longrightarrow_O$ if each derivative of P by $\longrightarrow_O$ has a finite number of *immediate* derivatives, i.e., for each $Q \in \mathcal{D}_O(P)$, it holds that the set $\{Q' | Q \xrightarrow{\alpha}_O Q'\}$ is finite.

A (*labelled*) *transition system* is a quadruple (S, T, R, s_0), where S is a set of states, T is a set of transition labels, $s_0 \in S$ is the initial state, and $R \subseteq S \times T \times S$ is a set of transitions. Given a process P and an operational semantics $\longrightarrow_O$, $O(P) = (\mathcal{D}_O(P), \mathcal{A}, \longrightarrow_O, P)$ is the transition system of P built by means of the relation $\longrightarrow_O$; for example, $S(P)$ is the standard transition system of P. Note that, with abuse of notation, we use $\longrightarrow_O$ for denoting both the operational semantics and the transition relation among the states of the transition system.

2.2 The mu-calculus

We use the modal mu-calculus [21] in a slightly extended form [27] as a branching temporal logic to express behavioural properties. The syntax of the extended mu-calculus is the following, where K ranges over sets of actions and Z ranges over variables:

$$\phi ::= \mathtt{tt} \mid \mathtt{ff} \mid Z \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid [K]\phi \mid \langle K \rangle \phi \mid \nu Z.\phi \mid \mu Z.\phi$$

A fixed point formula has the form $\mu Z.\phi$ ($\nu Z.\phi$) where μZ (νZ) *binds* free occurrences of Z in ϕ and an occurrence of Z is free if it is not within the scope of a binder μZ (νZ). A formula is *closed* if it contains no free variables. The formula $\mu Z.\phi$ is the least fixpoint of the recursive equation $Z = \phi$, while $\nu Z.\phi$ is the greatest one.

The verification of a formula ϕ by a (finite) term P is defined recursively in the following. In the definition, subformulae containing free variables are dealt with using *valuations*, i.e. functions ranged over by $\mathcal{V}$, which assign a subset $\mathcal{V}(Z)$ of processes in $\mathcal{E}$ to each variable Z. Moreover the notion of verification is given also with respect to a semantics $\longrightarrow_O$: the verification of ϕ by P and $\mathcal{V}$ is denoted by $P \models^O_{\mathcal{V}} \phi$. We assume that P is image finite with $\longrightarrow_O$. The transition system $O(P)$ verifies a formula ϕ, written $O(P) \models_{\mathcal{V}} \phi$, if and only if $P \models^O_{\mathcal{V}} \phi$, i.e. the initial state verifies ϕ by $\longrightarrow_O$ and $\mathcal{V}$.

$P \not\models^O_{\mathcal{V}} \mathtt{ff}$
$P \models^O_{\mathcal{V}} \mathtt{tt}$
$P \models^O_{\mathcal{V}} Z$ iff $P \in \mathcal{V}(Z)$
$P \models^O_{\mathcal{V}} \phi \wedge \psi$ iff $P \models^O_{\mathcal{V}} \phi \wedge P \models^O_{\mathcal{V}} \psi$
$P \models^O_{\mathcal{V}} \phi \vee \psi$ iff $P \models^O_{\mathcal{V}} \phi \vee P \models^O_{\mathcal{V}} \psi$
$P \models^O_{\mathcal{V}} [K]\phi$ iff $\forall P'. \forall \alpha \in K.$ if $P \xrightarrow{\alpha}_O P'$ then $P' \models^O_{\mathcal{V}} \phi$
$P \models^O_{\mathcal{V}} \langle K \rangle \phi$ iff $\exists P'. P' \models^O_{\mathcal{V}} \phi. \exists \alpha \in K. P \xrightarrow{\alpha}_O P'$
$P \models^O_{\mathcal{V}} \nu Z.\phi$ iff $P \models^O_{\mathcal{V}} \nu Z^n.\phi$ for all natural numbers n
$P \models^O_{\mathcal{V}} \mu Z.\phi$ iff $P \models^O_{\mathcal{V}} \mu Z^n.\phi$ for some natural number n

where $\nu Z^n.\phi$ and $\mu Z^n.\phi$ are defined as:

$$\begin{array}{ll} \nu Z^0.\phi = \mathtt{tt} & \mu Z^0.\phi = \mathtt{ff} \\ \nu Z^{n+1}.\phi = \phi\{\nu Z^n.\phi/Z\} & \mu Z^{n+1}.\phi = \phi\{\mu Z^n.\phi/Z\} \end{array}$$

where the notation $\phi\{\psi/Z\}$ indicates the substitution of ψ for every free occurrence of the variable Z in ϕ.

Note that closed formulae do not depend on valuations. Thus, in case of a closed formula ϕ we can simply write $P \models^O \langle K\rangle\phi$ in place of $P \models^O_{\mathcal{V}} \langle K\rangle\phi$. Moreover, the verification of a recursive formula by a term P is given considering natural numbers, instead of ordinals, since we consider only image finite terms [27].

In the sequel we will use the following abbreviations (where K range over sets of actions and $\mathcal{A}$ is the set of CCS actions):

$$[\alpha_1,\ldots,\alpha_n]\phi \stackrel{def}{=} [\{\alpha_1,\ldots,\alpha_n\}]\phi;\quad [-]\phi \stackrel{def}{=} [\mathcal{A}]\phi;\quad [-K]\phi \stackrel{def}{=} [\mathcal{A}-K]\phi$$

3 AN INFORMAL OVERVIEW OF THE APPROACH

In this section we present a brief overview of our approach, together with the problems it can solve. For this purpose, we use as an example the following CCS description of an automatic cash dispenser. The dispenser is able to perform two kinds of operations: to provide two different amounts of cash and to give information about a bank account. Each user of the cash dispenser owns a credit card with a personal code that must be supplied before requiring an operation. If the code is correctly inserted, the operation is accepted and executed after the return of the credit card; otherwise, the card is held and no operation is performed. In any case, the dispenser is able to go back to the state in which other requests can be accepted. After having correctly inserted the personal code, the user can ask either for one of two different amounts of money or for an account information. Then the dispenser returns the card and gives either the money or the requested information. In every case, the user must collect the item before the dispenser goes back to the initial state.

$$\begin{aligned} x \stackrel{def}{=} card.code.(\overline{right}.(cash.(&cash_1\_req.\overline{ret\_card}.\overline{cash_1}.collect.x + \\ &cash_2\_req.\overline{ret\_card}.\overline{cash_2}.collect.x) + \\ &account.\overline{ret\_card}.\overline{account\_info}.collect.x) + \\ &\overline{wrong}.\overline{hold}.x) \end{aligned}$$

Figure 1 shows $S(x)$, which has 13 states.

Now, let us suppose we want to verify the (mu-calculus) formula ψ_1 below:

$$\psi_1 = \nu Z.([\overline{cash_1}]\mathtt{ff} \wedge [-\overline{right}]Z)$$

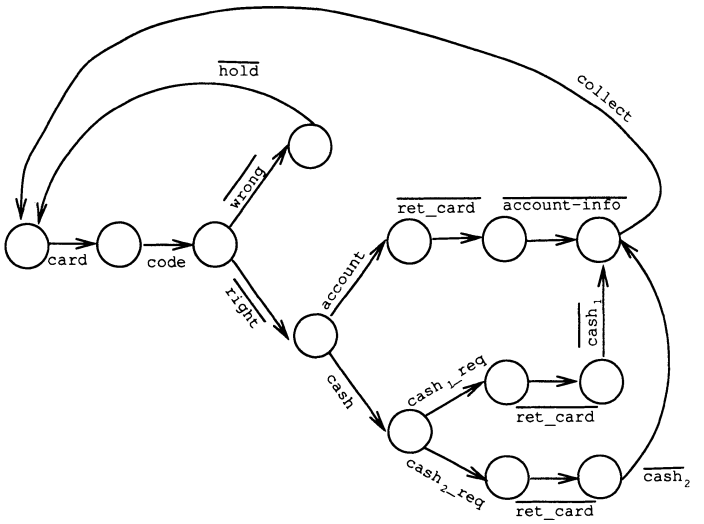

Figure 1

ψ_1 expresses the safety property: "after each action different from $\overline{right}$, an action $\overline{cash_1}$ cannot be performed".

It is easy to see that ψ_1 it satisfied by the transition system of Figure 1, but it has the same truth value if evaluated on the transition system of Figure 2(a), which is obtained from the transition system of Figure 1 by keeping only the transitions labelled by the actions $\overline{cash_1}$ and $\overline{right}$, and collapsing the states consequently. In fact we can note that, in order to check ψ_1, it is sufficient to observe only the part of the transition system containing these two actions. The problem we want to solve is to devise, given a formula, an automatic way for defining a suitable reduced system on which the formula has the same truth value of the complete system. In other words, given a formula ϕ, we look for a method to individuate those actions labelling transitions which do not alter the value of ϕ. Given such a set of actions, we can eliminate from the transition system the transitions labelled by them, and reduce the system consequently, still preserving the truth value of ϕ.

Consider again ψ_1. We note that the set of actions to be ignored does not coincide with the set of actions not occurring in the formula. In fact, this set contains only the action $\overline{right}$ (recall that $-a$ is a shorthand for $\mathcal{A} - a$), and generates the reduced transition system of Figure 2(b), if interpreted as the set of actions to be ignored. The formula ψ_1 is not satisfied by this transition systems, while it holds on the complete one.

It is important to note that it does not exist a mu-calculus formula expressing the above property and containing only the actions $cash_1$ and $\overline{right}$, which are the only ones relevant for proving the property. Intuitively, the "cycle" $\nu Z.(\ldots[-\overline{right}]Z)$ in ψ_1 means "go ahead over non-interesting actions"; thus,

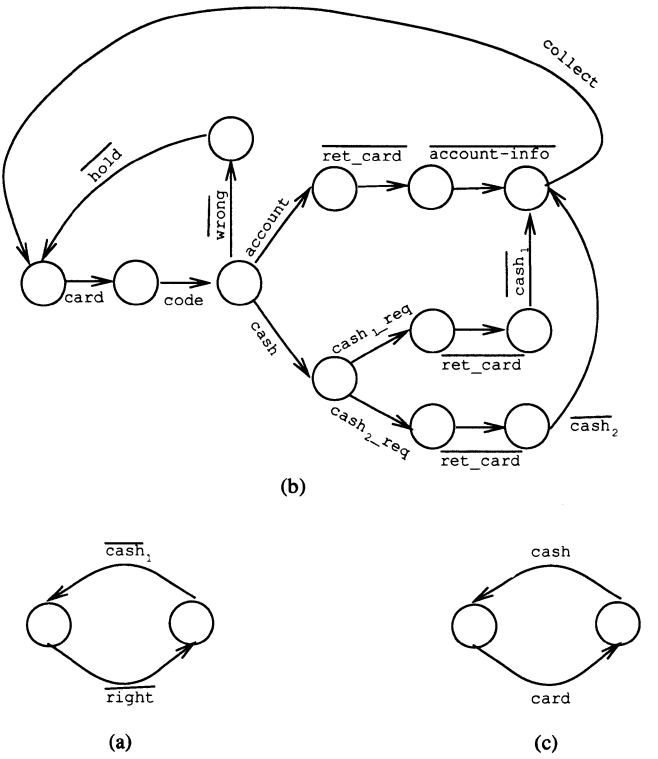

Figure 2

to express the fact that $\overline{right}$ is an interesting action, we need to mention all the other ones.

Consider now the following formula ψ_2, whose informal description is "it holds repeatedly that: there is a finite path leading to a $\overline{right}$ action and, after executing it, there is a finite path leading to a $\overline{cash_1}$ action".

$$\psi_2 = \nu Z.(\mu X.\langle -\overline{right}\rangle X \vee \langle \overline{right}\rangle \mathtt{tt}) \wedge (\nu W.[\overline{right}](\mu Y.\langle -cash_1\rangle Y \vee \langle cash_1\rangle Z) \wedge [-\overline{right}]W)$$

All the actions occur in this formula; nevertheless, it can be equivalently checked on the transition system of Figure 2(a). Thus, also in this case, all actions, apart from $\overline{right}$ and $\overline{cash_1}$, can be ignored. The above formulae seem to suggest that the interesting actions are only the ones occurring in the formula both in the form K and $-K$ inside the modal operators. It is sufficient the trivial formula $\psi_3 = [card]\langle cash\rangle \mathtt{tt}$ to realize that this is false. This formula is not satisfied by the transition system of Figure 1 but it is verified by the reduced transition system of Figure 2(c).

The above examples show that it does not exists an intuitive algorithm for extracting the set of actions to be ignored from a mu-calculus formula. On the other hand, they suggest the introduction of new modalities for expressing

properties, such that the actions which are relevant for proving a formula are the only ones explicitly mentioned by the modal operators occurring in the formula itself. For instance, we would like to express the property ψ_1 by a formula in which the only occurring actions are $\overline{cash_1}$ and $\overline{right}$. To this purpose, we define the (selective) modal operator $[K]_R$, where K and R are set of actions, such that $[K]_R\,\phi$ is verified by a process which, for every performance of a sequence of actions not belonging to $R \cup K$, followed by an action in K, evolves in a process obeying ϕ. With this new modal operator the property ψ_1 can be expressed by the formula: $\psi_{s1} = [\overline{cash_1}]_{\{\overline{right}\}}\mathtt{ff}$, in which the set of occurring actions is exactly $\{\overline{cash_1}, \overline{right}\}$. The new modality $\langle K\rangle_R\,\phi$ can be is defined analogously.

The idea of the selective mu-calculus is very simple although powerful. Formulae written using the new modalities can be checked equivalently, either on the complete transition system or on the one obtained by disregarding all the actions not occurring in the formula itself. A formula in selective mu-calculus corresponding to ψ_2 is $\psi_{s2} = \nu Z.\,\langle\overline{right}\rangle_\emptyset\,\mathtt{tt} \wedge [\overline{right}]_\emptyset\,\langle\overline{cash_1}\rangle_\emptyset\,Z$. The actions occurring in this formula "say" that it can be checked on the system of Figure 2(a).

For what regards ψ_3, we obtain the following formula in selective mu-calculus: $\psi_{s_3} = [card]_{\{\mathcal{A}-card\}}\langle cash\rangle_{\{\mathcal{A}-cash\}}\mathtt{tt}$. The occurring actions in this formula are the whole set $\mathcal{A}$; according to the fact that the formula is not checkable on the reduced system of Figure 2(c).

4 THE SELECTIVE MU-CALCULUS

The selective mu-calculus substitutes the modal operators $[K]$ and $\langle K\rangle$ with the selective operators $\langle K\rangle_R$ and $[K]_R$, with $R, K \subseteq \mathcal{A}$, the definition of which is the following:

$$
\begin{aligned}
P \models^O_V [K]_R\,\phi \quad &\text{iff} \quad \forall P'.\forall \delta \in (\mathcal{A} - (R \cup K))^*. \\
&\qquad \forall \alpha \in K.\text{if } P \xrightarrow{\delta}_O Q \xrightarrow{\alpha}_O P' \text{ then } P' \models^O_V \phi \\
P \models^O_V \langle K\rangle_R\,\phi \quad &\text{iff} \quad \exists P'.\exists \delta \in (\mathcal{A} - (R \cup K))^*. \\
&\qquad \exists \alpha \in K.P \xrightarrow{\delta}_O Q \xrightarrow{\alpha}_O P' \text{ and } P' \models^O_V \phi
\end{aligned}
$$

Informally, these new operators require that the formula ϕ is verified after the execution of an action of K, provided that it is not preceded by any action in $R \cup K$. More precisely:

> $[K]_R\,\phi$ is verified by a process which, for every performance of a sequence of actions not belonging to $R \cup K$, followed by an action in K, evolves to a process obeying ϕ.
> $\langle K\rangle_R\,\phi$ is verified by a process which can evolve to a process obeying ϕ after performing a sequence of actions not belonging to $R \cup K$, followed by an action in K.

The selective mu-calculus is equivalent to the mu-calculus. In fact it is easy to see that the standard mu-calculus operators can be defined by means of the selective operators subscribed by the whole set of actions $\mathcal{A}$:

$$[K]\phi = [K]_{\mathcal{A}}\,\phi \text{ and } \langle K\rangle\phi = \langle K\rangle_{\mathcal{A}}\,\phi$$

On the other hand, the selective operators can be expressed in standard mu-calculus as follows:

$$\langle K\rangle_R\,\phi = \mu Z.\langle K\rangle\phi \vee \langle -(R \cup K)\rangle Z \text{ and } [K]_R\,\phi = \nu Z.[K]\phi \wedge [-(R \cup K)]Z$$

Note that the mu-calculus formulae obtained by translating the selective mu-calculus operators have a structure recalling the one of formulae expressing, respectively, weak liveness and safety properties, as classified in [27].

Note also that, the translation from mu-calculus to selective mu-calculus produces formulae in which all the actions ($\mathcal{A}$) occur. This is not necessary in principle: we use this translation only to show how to pass, in a simple way, from one calculus to the other. Of course, it is possible to define more clever algorithms, which base the translation on the structure of mu-calculus formulae, such that the resulting formulae do not contain all the actions $\mathcal{A}$.

Given a set of actions $\rho \subseteq \mathcal{A}$ and a semantics $\longrightarrow_O$, we define a transition relation ignoring all actions in $\mathcal{A} - \rho$.

Definition 1 Given a set of actions $\rho \subseteq \mathcal{A}$ and an operational semantics $\longrightarrow_O$, we define the relation $\longrightarrow_{O^\rho}$ in the following way:

for each $\alpha \in \rho$ and $\delta \in (\mathcal{A} - \rho)^*$ $P \xrightarrow{\alpha}_{O^\rho} P' \equiv \exists Q.P \xrightarrow{\delta}_O Q \xrightarrow{\alpha}_O P'$.

By $P \xrightarrow{\alpha}_{O^\rho} P'$ we express the fact that it is possible to pass from P to P' (according to the operational semantics $\longrightarrow_O$) by performing a (possibly empty) sequence of actions not belonging to ρ and then the action α in ρ. Note that $\longrightarrow_{S^{\mathcal{A}}} = \longrightarrow_S$. Using the $\longrightarrow_{O^\rho}$ relation we now give the notions of ρ-bisimulation and ρ-equivalence between transition systems. Informally, two transition systems are ρ-equivalent iff they behave in the same way with respect to the actions in ρ.

Definition 2 (ρ-bisimulation, ρ-equivalence) Let $\rho \subseteq \mathcal{A}$ be a set of actions and $\longrightarrow_O$ and $\longrightarrow_\Omega$ two operational semantics. Let $O(P) = (\mathcal{S}_1, \mathcal{A}, \longrightarrow_O, P)$ and $\Omega(P') = (\mathcal{S}_2, \mathcal{A}, \longrightarrow_\Omega, P')$ the transition systems built for the terms P and P' using the two semantics.

- A *ρ-bisimulation*, $\mathcal{B}$, is a binary relation on $\mathcal{S}_1 \times \mathcal{S}_2$ such that $R\mathcal{B}Q$ implies:
  (i) $R \xrightarrow{\alpha}_{O^\rho} R'$ implies $Q \xrightarrow{\alpha}_{\Omega^\rho} Q'$ with $R'\mathcal{B}Q'$; and
  (ii) $Q \xrightarrow{\alpha}_{\Omega^\rho} Q'$ implies $R \xrightarrow{\alpha}_{O^\rho} R'$ with $R'\mathcal{B}Q'$
- $O(P)$ and $\Omega(P')$ are *ρ-equivalent* ($O(P) \approx_\rho \Omega(P')$) iff there exists a ρ-bisimulation $\mathcal{B}$ containing the pair (P, P').

To indicate that two CCS terms P and Q are ρ-equivalent with respect to an operational semantics $\longrightarrow_O$ (i.e. it occurs $O(P) \approx_\rho O(Q)$), we write $P \approx_\rho^O Q$.

Note that $\approx_{\mathcal{A}}^S$ coincides with Milner's strong equivalence; while $\approx_{\mathcal{L}}^S$, defined by considering only the visible actions, does not coincide with observational equivalence. In fact, τ actions are completely ignored by $\approx_{\mathcal{L}}^S$, but this does not occur in the case of observational equivalence. For example, the processes $a.nil + \tau.nil$ and $a.nil$ are $\mathcal{L}$-equivalent, while they are not observationally equivalent. On the other hand, $a.nil + a.(c.nil + \tau.nil)$ and $a.(c.nil + \tau.nil)$ are observationally equivalent, but they are not $\mathcal{L}$-equivalent. Actually, $\approx_{\mathcal{L}}^S$ is the same as the $\tau * \alpha$ *equivalence* defined in [14, 17], and implies the *safety equivalence* defined in [5].

Now we can formulate the main theorem of the paper, stating that two transition systems verify a formula ϕ of the selective mu-calculus iff there exists a ρ-bisimulation between them, where ρ contains the set of actions occurring in ϕ. This means that the set of formulae with occurring actions contained in ρ completely characterizes ρ-equivalence, as well as the set of all mu-calculus formulae characterizes strong equivalence [27].

Definition 3 (occurring actions) Given a formula ϕ of the selective mu-calculus, the set $\mathcal{C}(\phi)$ of the actions occurring in ϕ is inductively defined as follows:

- $\mathcal{C}(\mathtt{tt}) = \mathcal{C}(\mathtt{ff}) = \mathcal{C}(Z) = \emptyset$
- $\mathcal{C}(\langle K \rangle_R \phi) = \mathcal{C}([K]_R \phi) = K \cup R \cup \mathcal{C}(\phi)$
- $\mathcal{C}(\phi_1 \vee \phi_2) = \mathcal{C}(\phi_1 \wedge \phi_2) = \mathcal{C}(\phi_1) \cup \mathcal{C}(\phi_2)$
- $\mathcal{C}(\nu Z.\phi) = \mathcal{C}(\mu Z.\phi) = \mathcal{C}(\phi)$

Theorem 4 Let P and Q be two CCS terms and let $\longrightarrow_O$ and $\longrightarrow_\Omega$ be two operational semantics. Suppose that P is image finite by $\longrightarrow_O$ and Q is image finite by $\longrightarrow_\Omega$. For each $\rho \subseteq \mathcal{A}$:

$$O(P) \approx_\rho \Omega(Q) \qquad \text{if and only if}$$
$$P \models^O \phi \Leftrightarrow Q \models^\Omega \phi, \text{ for every } \phi \text{ such that } \mathcal{C}(\phi) \subseteq \rho.$$

Proof Sketch.
(only if) By natural induction on the depth of a formula ϕ of the selective mu-calculus, where the depth of ϕ is the number of nested selective operators $(\langle K \rangle_R$ and $[K]_R)$ in ϕ.
(if) By contradiction, i.e. by supposing that $O(P) \not\approx_\rho \Omega(Q)$ and by finding a formula ϕ such that $P \models^O \phi$ and $Q \not\models^\Omega \phi$.

Note that, as well as for mu-calculus and strong equivalence, the *only if* direction in the theorem above holds also for non-image finite terms, while the *if* direction holds only if the terms are image finite [2].

5 USING SELECTIVE MU-CALCULUS TO REDUCE STATE EXPLOSION

The selective mu-calculus has the property that, in each formula, the occurring actions are the only ones relevant to check the formula itself. In this section we discuss how state explosion can be reduced using selective mu-calculus. First of all we state the following proposition, relating transition systems obtained by using different operational semantics defined by O with different sets ρ of actions. We recall that, given a term P and a semantics $\longrightarrow_O$, $O^\rho(P)$ is the transition system generated by the operational semantics $\longrightarrow_{O^\rho}$.

Proposition 5 Given a term P and $\rho, \rho' \subseteq \mathcal{A}$, if $\rho \subseteq \rho'$, $O^\rho(P) \approx_\rho O^{\rho'}(P)$.
Proof Sketch. *By showing that* $\longrightarrow_{(O^{\rho'})^\rho} = \longrightarrow_{O^\rho}$.

If $O = S$ and $\rho' = \mathcal{A}$, the transition system generated by $\longrightarrow_{S^\rho}$ is ρ-equivalent to the one obtained by $\longrightarrow_{S^\mathcal{A}} \equiv \longrightarrow_S$, that is the standard transition system. As a consequence of the above proposition, a strategy to check a property ϕ on $S(P)$ may be that of checking it on $S^\rho(P)$, where $\rho = \mathcal{C}(\phi)$. In fact, in general, $S^\rho(P)$ is smaller than $S(P)$, even if it may be not the minimum one ρ-equivalent to $S(P)$. In order to furtherly reduce the state space, $S^\rho(P)$ can be minimized by known techniques finding the minimum transition system with respect to strong equivalence (see for example [12, 17]).

Example 6 Reconsider the CCS specification of the cash dispenser in Section 3 and let us express some other properties using the selective mu-calculus.

$\psi_1 = [\overline{hold}]_{\{\overline{wrong}\}}\mathtt{ff}$: "the card is not held if the wrong code is not inserted".

$\psi_2 = [\overline{right}]_\emptyset(\langle cash\rangle_\emptyset\,\mathtt{tt} \vee \langle account\rangle_\emptyset\,\mathtt{tt})$: "if the right code is inserted, it is possible to perform either a cash request or an account information".

$\psi_3 = \nu Z.[card]_\emptyset(Z \wedge [card]_{\{collect,\overline{hold}\}}\mathtt{ff})$: "a card can be inserted only if either the previously inserted card, if any, has been held or the previous operation, if any, has been successfully executed".

Each formula ψ_i, $i \in [1..3]$, can be checked on the transition systems $S^{\rho_i}(x)$ (reduced with respect to strong equivalence), where $\rho_i = \mathcal{C}(\psi_i)$: $\rho_1 = \{\overline{hold}, \overline{wrong}\}$, $\rho_2 = \{\overline{right}, cash, account\}$, $\rho_3 = \{card, collect, \overline{hold}\}$. Figure 3 shows $S^{\rho_i}(x)$ (reduced w.r.t. strong equivalence) for each i.

In order to effectively apply the above methodology to processes with any number of states, we need a tool able to build the reduced transition system $S^\rho(P)$, for a CCS term P and a set ρ of actions. We can simulate such a tool by using existing verification environments and standard notions of bisimulations. In fact, we can use the facilities for hiding actions (i.e. renaming some actions as τ), offered by most existing verification environments, and build

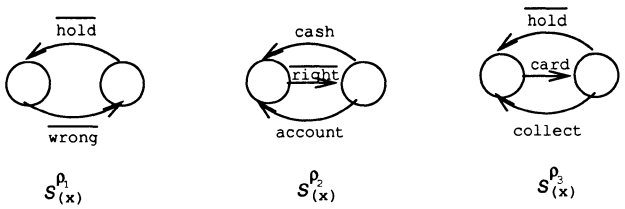

Figure 3

the minimum transition system with respect to a bisimulation ignoring τ actions. To experimentally evaluate the degree of reduction induced by selective mu-calculus, we used a known environment with its notions of bisimulation, i.e. the CADP environment [14, 17]. We applied the following methodology to build a reduced transition system for checking a formula with occurring actions ρ.

1. hide the actions in $\mathcal{A} - \rho$ in the specification, using the hiding facilities of CADP;
2. build the transition system with the `-imin` option of `aldebaran`, issuing $\tau^*\alpha$ equivalence reduction.

In order to show that the above strategy is correct, we state the following proposition:

Proposition 7 Let us denote by $H_\rho(P)$ the transition system obtained by $S(P)$ by substituting τ to all actions in $\mathcal{A} - \rho$. We have $H_\rho(P) \approx_\rho S(P)$. Moreover $H_\rho^{min}(P) \approx_\rho S(P)$, where $H_\rho^{min}(P)$ is the minimum transition system $\tau^*\alpha$ equivalent to $H_\rho(P)$.
Proof Sketch. *By Proposition 5 and by transitivity of* $\approx_\rho$.

Example 8 Let us consider the task scheduling system, taken from [23]: n processes wish to perform a task repeatedly, and a scheduler is required to ensure that they begin the task in cyclic order starting with the first process. The different task-performances need not exclude each other in time (for example the second process can begin before the first one finishes), but the scheduler is required to ensure that each agent finishes one performance before it begins the following. The action a_i signals to the i-th process that it can perform the task, whereas b_i signals its completion. The execution of each task is scheduled by a single process:

$$A \stackrel{def}{=} a.C \quad C \stackrel{def}{=} c.E \quad E \stackrel{def}{=} b.D + d.B \quad B \stackrel{def}{=} b.A \quad D \stackrel{def}{=} d.A$$

If we define $A_i \stackrel{def}{=} A[f_i]$, $D_i \stackrel{def}{=} D[f_i]$, etc., where $f_1 = [a_i/a,\ b_i/b,\ c_i/c,\ \overline{c_n}/d]$, and $f_i = [a_i/a,\ b_i/b,\ c_i/c,\ \overline{c_{i-1}}/d]$, for $1 < i \leq n$, an n-task scheduler is:

$$Sched_n \stackrel{def}{=} (A_1 \mid D_2 \mid \cdots \mid D_n)\backslash \tilde{c}$$

where $\tilde{c}$ denotes the set $\{c_1, \ldots, c_n\}$. The sort of the scheduler is $\{a_i, b_i \mid 1 \leq i \leq n\}$.
The properties we wish to prove about the scheduler are the following.

1. the start-task actions $a_1, \ldots, a_n$ are performed cyclically starting with a_1;
2. for each i, the start-task action a_i and the end-task action b_i are performed alternately.

A selective mu-calculus formula expressing (1) is:

$$\begin{aligned}\phi = \quad & \nu Z.\, [\tilde{a} - a_1]_{\{a_1\}}\, \mathtt{ff} \wedge [a_1]_{\emptyset} \\ & \big([\tilde{a} - a_2]_{\{a_2\}}\, \mathtt{ff} \wedge [a_2]_{\emptyset} \\ & \big([\tilde{a} - a_3]_{\{a_3\}}\, \mathtt{ff} \wedge \ldots \wedge [a_{n-1}]_{\emptyset} \\ & \big([\tilde{a} - a_n]_{\{a_n\}}\, \mathtt{ff} \wedge [a_n]_{\emptyset}\, Z\big) \ldots\big)\big)\end{aligned}$$

while the formulae expressing (2) are, for each $1 \leq i \leq n$, of the form

$$\psi_i = \nu Z.([b_i]_{\{a_i\}}\, \mathtt{ff} \wedge [a_i]_{\emptyset}([a_i]_{\{b_i\}}\, \mathtt{ff} \wedge [b_i]_{\emptyset}\, Z))$$

Note that the property expressed by ϕ is rather weak, since it implies that the a_i's are performed in cyclic order, but it does not imply that each a_i is ever executed.
Table 1 summarizes the experimental results obtained using CADP, showing the number of states of the standard transition systems and of the reduced ones, for some values of the number n of processes. In the table we use the following symbols:

- S_1: number of states of the standard transition system;
- S_2: number of states of the standard transition system minimized using the $\tau^*\alpha$ bisimulation;
- S_3: number of states of $H_{\rho}^{min}(Sched_n)$, where $\rho = \mathcal{C}(\phi) = \{a_1, \ldots, a_n\}$;
- S_4: number of states of $H_{\rho_i}^{min}(Sched_n)$, where $\rho_i = \mathcal{C}(\psi_i) = \{a_i, b_i\}$, for each $1 \leq i \leq n$.

Note that we obtain for the scheduler's example a reduction comparable to the one in [7]. The difference is that, while we derive the interesting actions from the formula, in [7] the hiding of the b_i actions is based on informal reasonings and consequently must be proved correct. Actually, our work can be seen as proving a general framework to extensively use practical techniques for process abstraction, driven by temporal logic formulae.
Finally, note that the above methodology cannot be used when $\tau \in \rho$; however, this is not a great limitation because in general it is not important to observe τ.

n	S_1	S_2	S_3	S_4
2	13	8	2	2
3	37	24	3	2
8	3073	2048	8	2
10	15361	10240	10	2

Table 1

6 CONCLUSION

In this paper we present a new temporal logic, the selective mu-calculus, with the property that the actions relevant to check a formula are only the ones occurring in the formula itself.

The degree of reduction we obtain depends on the actions occurring in a formula. This means that there are cases, i.e. when the actions occurring in the formula are almost the whole set $\mathcal{A}$ of actions, for which we do not obtain significant reductions. This occur when checking properties which must hold for every state of the transition system as, for example, deadlock-freeness. In fact our calculus deals with a specific kind of abstraction, namely deleting all paths in which some actions do not occur. Other kinds of abstractions were proposed in the literature, which are general abstractions or cope with a specific property, as, for example, deadlock freeness [7, 11, 28].

The selective mu-calculus is useful in practice because it allows the use a reduced transition system in property verification. Thus all the verification systems which base their behaviour on the analysis of transition systems can profit from the method. In particular, our approach can be integrated with an on-the-fly methodology [9, 15, 16, 19, 20], where on-the-fly means that the system is verified during its generation. Other approaches to model checking fall inside the automata-theoretic framework [4, 18, 30, 31], in which each temporal logic formula is associated with a (either word or tree) automaton accepting exactly all computations that satisfy the (negation of the) formula. To check whether a transition system satisfies a formula, a product is done between the transition system and the automaton describing the formula. Our approach can be also used in conjunction with this methodology, thus obtaining a more efficient verification.

REFERENCES

[1] A. Aziz, T.R. Shiple, V. Singhal, A.L. Sangiovanni-Vincentelli. *Formula-Dependent Equivalence for Compositional CTL Model Checking.* In Proceedings of Workshop on Computer Aided Verification (CAV'94),

LNCS 818, 1994. 324–337.

[2] R. Barbuti, N. De Francesco, A. Santone, G. Vaglini. *Formula-Based Reduction of Transition Systems.* Internal Report IR-3/97, Dipartimento di Ingegneria dell'Informazione, Univ. of Pisa.

[3] S. Bensalem, A. Bouajjani, C. Loiseaux, J. Sifakis. *Property Preserving Simulations.* In Proceedings of Workshop on Computer Aided Verification (CAV'92), LNCS 663, 1992. 260–273.

[4] O. Bernholtz, M.Y. Vardi, P. Wolper. *An Automata-Theoretic Approach to Branching-Time Model Checking.* In Proceedings of Workshop on Computer Aided Verification (CAV'94), LNCS 818, 1994. 142–155.

[5] A. Bouajjani, J.C. Fernandez, S. Graf, C. Rodriguez, J. Sifakis. *"Safety for Branching Time Semantics".* In Proceedings of the 18th International Colloquium on Automata, Languages and Programming. LNCS 510, 1991. 76–92.

[6] G. Bruns. *A Case Study in Safety-Critical Design.* In Proceedings of Workshop on Computer Aided Verification (CAV'92), LNCS 663, 1992. 220–233.

[7] G. Bruns. *A Practical Technique for Process Abstraction.* In Proceedings of International Conference on Concurrency Theory (CONCUR'93), LNCS 714, 1993. 37–49.

[8] R. Cleaveland, J. Parrow, B. Steffen. *The concurrency workbench: operating instructions.* Tech. Notes Sussex University, 1988.

[9] C. Courcoubetis, M. Vardi, P. Wolper, M. Yannakakis. *Memory Efficient Algorithms for the Verification of Temporal Properties.* In Workshop on Computer Aided Verification, DIMACS 90, June 1990.

[10] D. Dams, O. Grumberg, R. Gerth. *Generation of reduced models for checking fragments of CTL.* In Proceedings of Workshop on Computer Aided Verification (CAV'93), LNCS 697, 1993. 479–490.

[11] N. De Francesco, A. Santone, G. Vaglini. *A Non-Standard Semantics for Generating Reduced Transition Systems.* In Proceedings of LOMAPS'96, LNCS 1192, 1996. 370-387.

[12] R. De Simone, D. Vergamini. *Aboard AUTO.* INRIA Technical Report 111, 1989.

[13] R. De Simone, A. Ressouche. *Compositional semantics of ESTEREL and verification by compositional reductions.* In Proceedings of Workshop on Computer Aided Verification (CAV'94), LNCS 818, 1994. 441–454.

[14] J.C. Fernandez, A. Kerbrat, L. Mounier. *Symbolic Equivalence Checking.* In Proceedings of the 5th International Conference on Computer-Aided Verification, LNCS 697, 1993. 85-96.

[15] J.C. Fernandez, L. Mounier. *Verifying bisimulation on the fly.* In Third International Conference on Formal Description Techniques, FORTE'90, Madrid, November 1990.

[16] J.C. Fernandez, L. Mounier. *"On th Fly" Verification of Behavioural Equivalences and Preorders.* In Proceedings of the Third International

Conference on Computer-Aided Verification, LNCS 575, 1991. 181-191.

[17] J.C. Fernandez et al. *"CADP A Protocol Validation and Verification Toolbox"*. In Proceedings of the Third International Conference on Computer-Aided Verification, LNCS 1102, 1996. 437-440.

[18] T.A. Henzinger, O. Kupferman, M.Y. Vardi. *A Space-Efficient On-the-fly Algorithm for Real-Time Model Checking.* In Proceedings of International Conference on Concurrency Theory (CONCUR'96), LNCS 1119, 1996. 514–529.

[19] C. Jard, T. Jéron. *On-Line Model-Checking for Finite Linear Temporal Logic Specifications.* In International Workshop on Automatic Verification Methods for Finite State Systems, LNCS 407, 1989. 189–196.

[20] C. Jard, T. Jéron. *Bounded-memory Algorithms for Verification On-the-fly.* In Proceedings of the Third International Conference on Computer-Aided Verification, LNCS 575, 1991. 192-201.

[21] D. Kozen. *Results on the propositional mu-calculus.* Theoretical Computer Science, 27, (1983). 333–354.

[22] Y.S. Kwong. *On reduction of asynchronous systems.* Theoretical Computer Science 5, 1977. 25–50.

[23] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[24] D. Peled. *All from one, one for all, on model-checking using representatives.* In Proceedings of the Fifth International Conference on Computer-Aided Verification(CAV'93), LNCS 679, 1993. 409–423.

[25] D. Peled. *Combining Partial Order Reductions with On-the Fly Model-Checking.* In Proceedings of the 6th International Conference on Computer-Aided Verification(CAV'94), LNCS 818, 1994. 377–390.

[26] J. Sifakis. *Property Preserving Homomorphisms of Transition Systems.* In Logics of Programs, LNCS 164, 1983.

[27] C. Stirling. *An Introduction to modal and temporal logics for CCS* In Concurrency: Theory, Language, and Architecture, LNCS 391, 1989.

[28] A. Valmari. *A stubborn attack on state explosion.* In Proceedings of International Conference on Computer-Aided Verification (CAV'90), LNCS 531, 1990. 156–165.

[29] A. Valmari, M. Clegg. *Reduced Labelled Transition Systems Save Verification Effort.* In Proceedings of the International Conference on Concurrency Theory (CONCUR'91), LNCS, 1991. 526–540.

[30] M.Y. Vardi, P. Wolper. *An automata-theoretic approach to automatic program verification.* In Proceedings of the First Symposium on Logic Computer Science, Cambridge, 1986. 322–331,

[31] M.Y. Vardi, P. Wolper. *An automata-theoretic techniques for modal logics of programs.* Journal of Computer and System Science, 32(2):182-21, April 1986.