Formal Methods in System Design, 6, 1–35 (1995)

Property Preserving Abstractions for the Verification of Concurrent Systems *

CLAIRE LOISEAUX CLAIRE.LOISEAUX@IMAG.FR

SUSANNE GRAF SUSANNE.GRAF@IMAG.FR

JOSEPH SIFAKIS JOSEPH.SIFAKIS@IMAG.FR

AHMED BOUAJJANI AHMED.BOUAJJANI@IMAG.FR

SADDEK BENSALEM SADDEK.BENSALEM@IMAG.FR

VERIMAG , Rue Lavoisier, F-38330 Monbonnot, France*

Received October 1, 1992; Revised February 1, 1994

Editor: David Probst

Abstract. We study property preserving transformations for reactive systems. The main idea is the use of simulations parameterized by Galois connections (α, γ), relating the lattices of properties of two systems. We propose and study a notion of preservation of properties expressed by formulas of a logic, by a function α mapping sets of states of a system S into sets of states of a system S'. We give results on the preservation of properties expressed in sublanguages of the branching time μ-calculus when two systems S and S' are related via $\langle\alpha, \gamma\rangle$-simulations. They can be used to verify a property for a system by verifying the same property on a simpler system which is an abstraction of it. We show also under which conditions abstraction of concurrent systems can be computed from the abstraction of their components. This allows a compositional application of the proposed verification method.

This is a revised version of the papers [2] and [16]; the results are fully developed in [27].

Keywords: abstract interpretation, simulation, property preservation, model-checking.

1. Introduction

The growing complexity of distributed and reactive systems requires rigorous development methodologies and automatic verification techniques. A well-known limitation of automatic verification techniques is their applicability only to relatively small finite state programs because of the exponential blow-up of the size of the models that have to be constructed for their application. Many techniques have been developed in order to push further the limits of model-checking. One of them consists in using *property preserving abstractions*: Given a program and a property to be verified, find a (simpler) abstract program such that the satisfaction on the

*This work was partially supported by ESPRIT Basic Research Action "REACT"

*Verimag is a joint laboratory of CNRS, Institut National Polytechnique de Grenoble, Université J. Fourier and Verilog SA associated with IMAG

abstract program implies the satisfaction on the initial program, called concrete program in this context. An important point is, given a concrete program, how to *construct* an abstract program that is both, simple enough in order to be verified by available tools, and that still contains enough relevant details for the satisfaction of the considered properties.

The framework of abstract interpretation (see for example [7], [8]) addresses exactly this problem. Programs are represented by functions F on some lattice of properties. Given some abstract lattice of properties and a pair of functions (α, γ), forming a Galois connection [33] from the concrete to the abstract lattice, a function G on the abstract lattice is an abstraction of F if $\alpha \circ F \circ \gamma \subseteq G$ holds. This guarantees that greatest and least fixpoints of G represent upper approximations of corresponding fixpoints of F. Until recently, this approach has only been applied for the verification of invariance properties of sequential programs. However, in [38], [39], the idea of abstract interpretation has been applied to programs represented by transition systems, where the lattice of properties is the powerset of states. There, results showing preservation of fragments of CTL [9] from the abstract to the concrete system have been given.

In the framework of process algebras, the problem of property preserving preorders and equivalences has also been widely studied. In this framework, the notions of abstractions are generally defined in terms of variants of simulation [29] and bisimulation [30]; the problem of the construction of abstract programs has only been addressed for notions of abstractions defined by equivalences.

In the linear semantics framework, the intuitive notion of abstraction is inclusion (respectively equality) of observable computation sequences (see for example in [25], [1], [28]). However, this notion of abstraction does not directly induce a way of computing an abstract program for a given concrete program and observability criterion.

Here, we take up again the approach followed in [38], [39]. We define a notion of abstraction on transition systems as a simulation parameterized by Galois connections (α, γ). We show that the notion of abstraction induced by $\langle \alpha, \gamma \rangle$-simulation coincides exactly with the notion of abstraction defined by simulation in the sense of Milner [29], parameterized by the relation ρ corresponding to the Galois connection (α, γ).

Then, we give preservation results for fragments of a future and past version of the branching time μ-calculus defined in [24] for the following notion of property preservation : an arbitrary function α from the powerset of the states of a transition system S_1 to the powerset of the states of a transition system S_2 *preserves* a property f if for any state of S_1 which satisfies f, all the states of S_2 in its image also satisfy f. If the converse also holds, then we say that α *strongly preserves* f. A preservation result of particular practical interest, says that if two systems are related via $\langle \alpha, \gamma \rangle$-simulation, then all formulas of the μ-calculus using no negation and only universal quantification over computation sequences (called $\Box L_\mu$) are preserved by $\widetilde{\gamma}$ from the abstract to the concrete system (where $\widetilde{\gamma}$ is the dual of γ).

These preservation results generalize results given in [10] where this problem is studied in the particular case where the property preserving function α defines a structure homomorphism from the concrete to abstract system.

Our preservation results together with the fact that, given some concrete system and some connection (α, γ), an abstract system can be computed, allow the use of the following verification method. In order to verify a property — expressed as a formula f of $\Box L_\mu$ — on a system S, provide a connection (α, γ) between the powersets of concrete and abstract states, compute the associated abstract system S_A and verify f on S_A. If f holds on S_A, it also holds on S.

Finally, we give a result concerning compositionality of simulation over parallel composition, which is important for the application of this method in practice. From a practical point of view, there are two reasons for building an abstract program of a composed system by composition of abstractions of its components. It is easier to define connections (α, γ) separately for each component than for the compound system; proceeding this way allows also to avoid building a representation of the global transition system associated with the composed system. As well for synchronous as for asynchronous parallel composition (allowing shared variables between components), we give compositionality results, that means rules, allowing to deduce $\langle \alpha, \gamma \rangle$-simulation for a compound system from $\langle \alpha_i, \gamma_i \rangle$-simulations for its components, where $\langle \alpha, \gamma \rangle$ is expressed in terms of $\langle \alpha_i, \gamma_i \rangle$.

The paper is organized as follows. In Section 2, we give some notations and recall the definition of Galois connections and some interesting properties of them. In Section 3, the definition of $\langle \alpha, \gamma \rangle$-simulation is given. We show that this notion coincides with the usual notion of simulation. In Section 4, we define a notion of *abstract program* obtained from a given function α or its associated relation ρ. Section 5 presents the notion of property preservation and general results allowing to prove that a function preserves the validity of formulas of a given language. Section 6 gives results concerning the preservation of fragments of the μ-calculus when transition systems are related via $\langle \alpha, \gamma \rangle$-simulation. Section 7, gives results concerning the compositionality of simulation with respect to parallel composition. Finally, Appendix A contains some technical proofs.

2. Preliminary definitions

In Section 3, we study the relationship between the notions of abstraction in the frameworks of process algebras and of abstract interpretation. In this section, we define the basic notions, necessary for this comparison. In process algebras programs are modeled as transition systems, that means as binary relations on the set of states. In the framework of abstract interpretation, programs are represented by predicate transformers, i. e., functions transforming sets of states into sets of states.

With any transition relation R can be associated different predicate transformers, the forward and backward image functions, which we denote here by $pre[R]$, respectively $post[R]$. In the abstract interpretation framework, the notion of abstraction is based on the existence of a Galois connection between the lattices of properties. We recall here the definition of Galois connection and some well-known properties concerning them, which are used in the proofs later on.

2.1. Transition systems and predicate transformers

Definition 1 *(Transition systems)*
A transition system *is a pair $S = (Q, R)$, where Q is a set of states and R is a transition relation on Q ($R \subseteq Q \times Q$).*

Notation 1 *We adopt the following conventions and notations:*

- *We identify a unary predicate on Q with its characteristic set since the lattice of unary predicates is isomorphic to 2^Q. Thus, for a unary predicate P and a state $q \in Q$, the notations $P(q) = true$, $P(q)$ and $q \in P$ are equivalent.*
- *We denote by Id_Q the* identity *function on 2^Q.*
- *Given two relations $R \subseteq Q \times Q'$ and $S \subseteq Q' \times Q''$ and two functions $f : Q \rightarrow Q'$ and $g : Q' \rightarrow Q''$, then denote the composition of the relations R and S by RS and the composition of the functions f and g by $g \circ f$, respectively $g(f(q))$ if $g \circ f$ is applied to some argument $q \in Q$.*

In the sequel, we consider always properties to be state properties, i. e., interpreted as a set of states (or a corresponding unary predicate). Therefore, in the sequel *property lattice* is always the same as *powerset on the set of states.*

Definition 2 *(The predicate transformers pre and post)*
Given a relation $\rho \subseteq Q_1 \times Q_2$, we define $pre[\rho] : 2^{Q_2} \rightarrow 2^{Q_1}$ and $post[\rho] : 2^{Q_1} \rightarrow 2^{Q_2}$ by,

$$pre[\rho] \stackrel{def}{=} \lambda X.\ \{q_1 \in Q_1\ :\ \exists q_2 \in X.\ \ q_1\ \rho\ q_2\}$$
$$post[\rho] \stackrel{def}{=} \lambda X.\ \{q_2 \in Q_2\ :\ \exists q_1 \in X.\ \ q_1\ \rho\ q_2\}$$

That means, for ${Q_2}'\subseteq Q_2$, $pre[\rho]({Q_2}')$ represents the set of *predecessors* of the states of ${Q_2}'$ via the relation ρ and for ${Q_1}'\subseteq Q_1$, $post[\rho]({Q_1}')$ represents the set of *successors* of the states of ${Q_1}'$ via ρ. Notice also that we have $post[\rho] = pre[\rho^{-1}]$.

The following propositions give some useful results concerning the predicate transformers *pre* and *post* which can for example be found in [39].

Proposition 1 *For any relation ρ from a set Q_1 to a set Q_2 ($\rho \subseteq Q_1 \times Q_2$), we have:*

1. $pre[\rho](\emptyset) = \emptyset$,

2. For any X_1, X_2 subsets of Q_2, $pre[\rho](X_1 \cup X_2) = pre[\rho](X_1) \cup pre[\rho](X_2)$,

Notation 2 *(Dual of a function)*
We denote by $\widetilde{\alpha}$ the dual of a function $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$ that is

$$\widetilde{\alpha} \stackrel{def}{=} \lambda X . \overline{\alpha(\overline{X})}.$$

Proposition 2 *Let be $\rho \subseteq Q_1 \times Q_2$ and $\sigma \subseteq Q_2 \times Q_3$. Then,*

- $pre[\rho\sigma] = pre[\rho] \circ pre[\sigma]$,
- $\widetilde{pre}[\rho\sigma] = \widetilde{pre}[\rho] \circ \widetilde{pre}[\sigma]$,
- $post[\rho\sigma] = post[\sigma] \circ post[\rho]$,
- $\widetilde{post}[\rho\sigma] = \widetilde{post}[\sigma] \circ \widetilde{post}[\rho]$.

2.2. Galois connections

We give hereafter the definition of Galois connections and some useful well-known results about them. More information can, e. g., be found in [33], [37].

Definition 3 *(Connections)*
Let Q_1 and Q_2 be two sets of states. A connection from 2^{Q_1} to 2^{Q_2} is a pair of monotonic functions (α, γ), where $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$ and $\gamma : 2^{Q_2} \rightarrow 2^{Q_1}$, such that $Id_{Q_1} \subseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \subseteq Id_{Q_2}$.

Proposition 3 *For any connection (α, γ) from 2^{Q_1} to 2^{Q_2}, we have,*

- $\alpha(\emptyset) = \emptyset$,
- α *distributes over* $\cup$ *and* γ *distributes over* $\cap$,
- $\alpha \circ \gamma \circ \alpha = \alpha$, *and* $\gamma \circ \alpha \circ \gamma = \gamma$,
- $(\widetilde{\gamma}, \widetilde{\alpha})$ *is a connection from* 2^{Q_2} *to* 2^{Q_1}.
- $\forall Q \subseteq Q_1, Q' \subseteq Q_2 \,.\, \alpha(Q) \subseteq Q'$ *iff* $Q \subseteq \gamma(Q')$.

Proposition 4 *Let $F : 2^{Q_1} \rightarrow 2^{Q_1}$ and $G : 2^{Q_2} \rightarrow 2^{Q_2}$ be two functions and (α, γ) a connection from 2^{Q_1} to 2^{Q_2}. Then,*

$$\alpha \circ F \circ \gamma \subseteq G \text{ if and only if } F \subseteq \gamma \circ G \circ \alpha$$

Proposition 5 *For any connection (α, γ) from 2^{Q_1} to 2^{Q_2}, we have,*

- $\gamma = \lambda Y . \bigcup \{X \in 2^{Q_1} \;:\; \alpha(X) \subseteq Y\}$,

- $\alpha = \lambda X. \bigcap \{Y \in 2^{Q_2} \ : \ X \subseteq \gamma(Y)\}$.

That means that α and γ determine each other in a unique manner. These characterizations allow to deduce the following two propositions showing the links between the connections from 2^{Q_1} to 2^{Q_2} and the binary relations from Q_1 to Q_2.

Proposition 6 *(Connections generated by a binary relation on states)*
If $\rho \subseteq Q_1 \times Q_2$, then the pair $(post[\rho], \widetilde{pre}[\rho])$ is a connection from 2^{Q_1} to 2^{Q_2} and $(pre[\rho], \widetilde{post}[\rho])$ is a connection from 2^{Q_2} to 2^{Q_1}.

Proposition 7 *(Relations induced by connections)*
If (α, γ) is a connection from 2^{Q_1} to 2^{Q_2}, then there exists a unique relation $\rho \subseteq Q_1 \times Q_2$ such that $\alpha = post[\rho]$ and $\gamma = \widetilde{pre}[\rho]$.

Proof: Let (α, γ) be a connection from 2^{Q_1} to 2^{Q_2}. Consider the relation ρ defined by $(q_1, q_2) \in \rho$ if and only if $q_2 \in \alpha(q_1)$. Since $\alpha(\emptyset) = \emptyset$ and α distributes over $\cup$ (Proposition 3), we have $\alpha = post[\rho]$.

Furthermore, by the Proposition 5, we have $\gamma = \lambda Y. \bigcup \{X \in 2^{Q_1} \ : \ \alpha(X) \subseteq Y\}$, and as α distributes over $\cup$, we can write $\gamma = \lambda Y. \{q \in Q_1 \ : \ \alpha(\{q\}) \subseteq Y\}$. Now, since $\alpha = post[\rho]$, it is easy to deduce that $\gamma = \widetilde{pre}[\rho]$. ∎

Proposition 8 *If (α, γ) is a connection from 2^{Q_1} to 2^{Q_2} and (α', γ') is a connection from 2^{Q_2} to 2^{Q_1}, then we have,*

1. *$Id_{Im(\widetilde{\gamma})} \subseteq \widetilde{\gamma} \circ \alpha$ and $Id_{Im(\alpha)} \subseteq \alpha \circ \widetilde{\gamma}$,*
2. *$\widetilde{\gamma}' \circ \widetilde{\gamma} \circ \widetilde{\gamma}' = \widetilde{\gamma}'$ if and only if $\alpha' \circ \alpha \circ \alpha' = \alpha'$.*

Proof: Consider the relation $\rho \subseteq Q_1 \times Q_2$ such that $\alpha = post[\rho]$ and $\gamma = \widetilde{pre}[\rho]$, which exists by Proposition 7.

Now, it is easy to see that $Id_{Q_1} \subseteq pre[\rho] \circ post[\rho]$ for any $\rho \subseteq Q_1 \times Q_2$ that is total on Q_1 and $Id_{Q_2} \subseteq post[\rho] \circ pre[\rho]$ for any $\rho \subseteq Q_1 \times Q_2$ that is total on Q_2.

By Proposition 7, the equation $\widetilde{\gamma}' \circ \widetilde{\gamma} \circ \widetilde{\gamma}' = \widetilde{\gamma}'$ is equivalent to $pre[\rho'] \circ pre[\rho] \circ pre[\rho'] = pre[\rho']$ for some appropriate relations ρ, ρ'. By Proposition 2, this is equivalent to $pre[\rho' \rho \rho'] = pre[\rho']$, that is $\rho' = \rho' \rho \rho'$. Symmetrically, $\rho' = \rho' \rho \rho'$ is equivalent to $post[\rho'] = post[\rho' \rho \rho']$, that is to $post[\rho'] = post[\rho'] \circ post[\rho] \circ post[\rho']$, i. e., $\alpha' = \alpha' \circ \alpha \circ \alpha'$. ∎

3. Simulations

In this section, we define a notion of simulation based on Galois connections (α, γ), called $\langle \alpha, \gamma \rangle$-simulation. Its definition is inspired by the notion of *abstract interpretation* in the sense of Cousot [7], [8]. There, a program is represented by a function F mapping properties into properties. A function G, mapping abstract properties into abstract properties, is an *abstraction* of F if there exists a connection (α, γ) from the concrete to abstract lattice of properties, such that $\alpha \circ F \circ \gamma \subseteq G$.

In our framework, where a program is a transition system $S_1 = (Q_1, R_1)$, a possible choice for the function F is taking one of the predicate transformers associated with the transition relation R. We consider that the expressions S_A *is an abstraction of* S and S *simulates* S_A are equivalent. We show that the notion of abstraction induced by the choice $F = pre[R_1]$ coincides with the notion of abstraction induced by simulation in the sense of Milner [29] which is used in the framework of process algebras.

3.1. Simulations induced by connections

First, we define simulation (and bisimulation) parameterized by a connection (α, γ) relating the property lattices of two transition systems $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$, i. e., a connection from 2^{Q_1} to 2^{Q_2}.

Definition 4 *($\sqsubseteq_{\langle\alpha,\gamma\rangle}$ and $\simeq_{\langle\alpha,\gamma\rangle}$)*
Let $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems and (α, γ) be a connection from 2^{Q_1} to 2^{Q_2}. Define,

- *$S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ if and only if $\alpha \circ pre[R_1] \circ \gamma \subseteq pre[R_2]$,*
- *$S_1 \simeq_{\langle\alpha,\gamma\rangle} S_2$ if and only if $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ and $S_2 \sqsubseteq_{\langle\widetilde{\gamma},\widetilde{\alpha}\rangle} S_1$.*

If $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$, we say that S_1 $\langle\alpha,\gamma\rangle$-simulates S_2 or S_2 is an $\langle\alpha,\gamma\rangle$-abstraction of S_1. A useful dual condition for the definition of $\langle\alpha,\gamma\rangle$-simulation can be deduced from Proposition 4.

3.2. Relating $\langle\alpha,\gamma\rangle$-simulation and behavioural simulation

We recall first the definitions of behavioural simulation and bisimulation in the sense of Milner which are based on a binary relation ρ between the sets of states Q_1 and Q_2. In Propositions 9 and 10 we show that these two notions of simulation coincide.

Definition 5 *($\sqsubseteq_\rho$ and $\simeq_\rho$)*
Let $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems and ρ be a relation from Q_1 to Q_2 ($\rho \subseteq Q_1 \times Q_2$). Define,

- *$S_1 \sqsubseteq_\rho S_2$ if and only if ${R_1}^{-1}\ \rho \subseteq \rho\ {R_2}^{-1}$,*
- *$S_1 \simeq_\rho S_2$ if and only if $S_1 \sqsubseteq_\rho S_2$ and $S_2 \sqsubseteq_{\rho^{-1}} S_1$.*

If $S_1 \sqsubseteq_\rho S_2$, we say that S_1 ρ-simulates S_2 or S_2 is a ρ-abstraction of S_1.

S_1 simulates (respectively, bisimulates) the system S_2 if *there exists* a relation ρ such that $S_1 \sqsubseteq_\rho S_2$ (respectively $S_1 \simeq_\rho S_2$). We show now that $\langle\alpha,\gamma\rangle$-simulation and ρ-simulation coincide.

Proposition 9 *(From $\sqsubseteq_{\langle\alpha,\gamma\rangle}$ to $\sqsubseteq_\rho$)*
Let $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems. For any relation $\rho \subseteq Q_1 \times Q_2$, there exists a connection (α, γ) from 2^{Q_1} to 2^{Q_2} such that $S_1 \sqsubseteq_\rho S_2$ if and only if $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$.

Proof: We show that the intended connection is $(post[\rho], \widetilde{pre}[\rho])$ (by Proposition 6, this pair is indeed a connection). Suppose that $S_1 \sqsubseteq_{\langle post[\rho], \widetilde{pre}[\rho]\rangle} S_2$, i. e.,

$$post[\rho] \circ pre[R_1] \circ \widetilde{pre}[\rho] \subseteq pre[R_2].$$

Then, as $post[\rho]$ is monotonic and $Id_{Q_1} \subseteq \widetilde{pre}[\rho] \circ post[\rho]$, we obtain,

$post[\rho] \circ pre[R_1] \circ \widetilde{pre}[\rho] \circ post[\rho] \subseteq pre[R_2] \circ post[\rho]$ which implies
$post[\rho] \circ pre[R_1] \subseteq pre[R_2] \circ post[\rho]$ which is equivalent to ${R_1}^{-1}\, \rho \subseteq \rho\, {R_2}^{-1}$.

It can be shown in a similar way that the converse also holds. This proves,

$$S_1 \simeq_{\langle post[\rho], \widetilde{pre}[\rho]\rangle} S_2 \text{ if and only if } S_1 \simeq_\rho S_2. \qquad \blacksquare$$

Proposition 10 *(From $\sqsubseteq_\rho$ to $\sqsubseteq_{\langle\alpha,\gamma\rangle}$)*
Let $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems. For any connection (α, γ) from 2^{Q_1} to 2^{Q_2} there exists a relation $\rho \subseteq Q_1 \times Q_2$ such that

$$S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2 \text{ if and only if } S_1 \sqsubseteq_\rho S_2.$$

Proof: Direct from Propositions 7 and 9. $\blacksquare$

This result clarifies the relationship between the approach of abstract interpretation and that chosen in the framework of process algebra. In fact, the notion of abstraction in the case where program models are transition systems is the same. Therefore, we do not distinguish in the sequel between simulations parameterized by relations and those parameterized by connections; in any context we use the notion which allows to present the results in the simplest way.

4. Computing program abstractions

In the framework of process algebra and of program refinement, the notion of simulation is in general used in order to decide for two given programs if one of them simulates the other. But our aim is, given a program P and a relation ρ relating concrete and abstract states, to *construct* an abstract program P_A such that P ρ-simulates P_A. Obviously, there are many programs which are ρ-abstractions of

P. In particular the program *Chaos* defined by the universal transition relation on the abstract set of states is a trivial ρ-abstraction of any P. We are interested in an abstract program satisfying — for a given ρ — as many properties as possible, i.e. which is as close as possible to the concrete program.

In our framework, where P is represented by a transition system $S = (Q, R)$ the abstract program must also be representable by some transition relation of the form $S_A = (Q_A, R_A)$, where Q_A is the set of abstract states. In this case the obvious minimal function $post[\rho] \circ pre[R] \circ \widetilde{pre}[\rho]$ — obtained from the definition of simulation — does not necessarily correspond to a solution, that means a function of the form $pre[R_A]$ for some transition relation R_A.

It is easy to see that in general, there may exist several *minimal* ρ-abstractions of S. In Section 4.1, we define first the criterium of *faithfulness* which is satisfied by all transition systems on Q_A which are bisimilar to any smaller (in the sense of inclusion) ρ-abstractions of S. Using the results of Section 5, we will see that faithful abstractions are the set of abstract programs which satisfy all properties which are possibly satisfied by any ρ-abstraction of S and which are *preserved* from S_A to S.

We will see that the abstract program defined by $S_\rho = (Q_A, R_\rho)$ with $R_\rho = \rho^{-1}\ R\ \rho$ is a faithful abstraction if ρ is total and moreover $\rho = \rho\ \rho^{-1}\ \rho$ holds. In the case that ρ is a total function, $pre[\rho] = \widetilde{pre}[\rho]$ holds, which trivially implies that S_ρ is the least abstraction. Then, ρ defines a structure homomorphism from S to S_ρ; this case has been widely studied in the literature (see for example in [25], [10]).

S_ρ is induced in an obvious manner by a slightly stronger notion of simulation than $\sqsubseteq_\rho$ which we denote by $\preceq_\rho$. Under some conditions $\preceq_\rho$ coincides with the notion of *forward and backward simulation* for which we obtain stronger preservation results than for $\sqsubseteq_\rho$.

In Section 4.2, we show how S_ρ can be computed if transition relations as well as abstraction relations ρ are represented by predicates over sets of program variables and illustrate this on a small example.

4.1. Faithful abstractions

Definition 6 *(Faithful abstractions)*
Given $S = (Q, R)$ and $\rho \subseteq Q \times Q_A$, we say that $S_A = (Q_A, R_A)$ is a faithful *abstraction of S via ρ if $S \sqsubseteq_\rho S_A$ and $\forall S' = (Q_A, R')\ .\ S \sqsubseteq_\rho S'$ and $R' \subseteq R_A$ implies $\exists \rho' \subseteq Q_A \times Q_A\ .\ S_A \simeq_{\rho'} S'$.*

Notation 3 *(The system S_ρ)*
Given $S = (Q, R)$ and $\rho \subseteq Q \times Q_A$, total on Q, we define $S_\rho = (Q_A, R_\rho)$ where $R_\rho = \rho^{-1}\ R\ \rho$ (or equivalently, $pre[R_\rho] = post[\rho] \circ pre[R] \circ pre[\rho]$).

Proposition 11 *Let $S = (Q, R)$ be a transition system and $\rho \subseteq Q \times Q_A$.*

- *If ρ is total on Q, then $S \sqsubseteq_\rho S_\rho$.*

- *If furthermore $\rho = \rho\ \rho^{-1}\ \rho$, then S_ρ is a faithful abstraction of S via ρ.*
- *If ρ is a (total) function, then S_ρ is* the *least ρ-abstraction of S.*

Proof: The first and the third items follow directly from the fact that $\widetilde{pre}[\rho] \subseteq pre[\rho]$ if ρ is total on Q (respectively $\widetilde{pre}[\rho] = pre[\rho]$ if ρ is a function). For the second item, we show that for any transition system $S_A = (Q_A, R_A)$ such that $S \sqsubseteq_\rho S_A$ and $R_A \subseteq R_\rho$, we have $S_A \simeq_{\rho^{-1}\rho} S_\rho$, the proof of which is given in the Appendix A.1.
■

Notice that $\rho = \rho\rho^{-1}\rho$ if and only if any two states of Q having a common successor by ρ, have the *same* successors by ρ. This means that ρ defines a function from the partition on Q induced by $\rho\rho^{-1}$ into the partition of Q_A induced by $\rho^{-1}\rho$. There exist examples of interesting abstraction relations ρ such that ρ is not a function. If $\rho = \rho\rho^{-1}\rho$ does not hold, then S_ρ is not necessarily faithful, and in [12] is given a way to compute faithful abstractions.

S_ρ is induced by a slightly stronger notion of simulation than $\sqsubseteq_\rho$ (respectively $\sqsubseteq_{\langle\alpha,\gamma\rangle}$) which coincides with the notion of *forward and backward* simulation used, e. g. in [21], [22] if ρ is total.

Definition 7 *($\preceq_\rho$ and $\preceq_{\langle\alpha,\gamma\rangle}$)*
Let $S = (Q, R)$ and $S_A = (Q_A, R_A)$ be transition systems, and $\rho \subseteq Q \times Q_A$ total on Q and (α, γ) a total connection from 2^Q to 2^{Q_A}. Then,

- *$S \preceq_\rho S_A$ if and only if $\rho^{-1}\ R\ \rho \subseteq R_A$*
- *$S \preceq_{\langle\alpha,\gamma\rangle} S_A$ if and only if $\alpha \circ pre[R] \circ \widetilde{\gamma} \subseteq pre[R_A]$*

Lemma 1 *(Characterization of $\preceq_\rho$)*
Let $S = (Q, R)$ and $S_A = (Q_A, R_A)$ be transition systems, and $\rho \subseteq Q \times Q_A$ total on Q; denote $S^{-1} = (Q, R^{-1})$ and analogously for S_A. Then,

$S \preceq_\rho S_A$ if and only if $S \sqsubseteq_\rho S_A$ and $S^{-1} \sqsubseteq_\rho S_A^{-1}$.

4.2. Symbolic computation of program abstractions

Now, we consider the particular case where transition relations and abstraction relations are represented by predicates over program variables. The sets of states Q are the Cartesian product of the domains of a tuple of program variables. For example, if $X = (x, y)$, then we have, $Q = Dom(X) = Dom(x) \times Dom(y)$.

Then, binary relations on $Dom(X)$ can be represented by binary predicates of the form $R(X, X')$ where $X' = (x', y')$ is a "copy" of X, i.e., $Dom(X) = Dom(X')$. X encodes the source state and X' the target state of any transition in R. For example., if $Dom(x) = \mathcal{N}$ and $Dom(y) = Bool$, then $R = y \wedge (x' = x + 1)$

represents the transition relation relating any $(n, true) \in \mathcal{N} \times Bool$ with $(n+1, b')$ where b' may take any boolean value as y' is not constraint in the expression R. This approach is used, e. g., in [26], [35]. In the same way a relation ρ from $Dom(X)$ to $Dom(X_A)$ is represented as a binary predicate of the form $\rho(X, X_A)$.

In this setting, operations on sets (respectively relations) are expressed by logical connectives. For example, the fact that a relation R_1 is included in R_2 is expressed by $R_1 \Rightarrow R_2$ and $R_1 \wedge R_2$ represents the intersection of R_1 and R_2 if they are defined on the same set of variables.

We consider that a program is a family of transition relations represented by sets of binary predicates on the same tuple of variables, $S = \{R_i(X, X')\}_{i \in I}$ where $i \in I$ are used as labels (names) for synchronization purposes in parallel composition in Section 7.

Then, given an abstraction relation $\rho(X, Y)$, where Y is a tuple of abstract variables, the abstraction S_ρ of S is computed as

$$S_\rho = \{\exists X \exists X' \ . \ \rho(X, Y) \wedge \rho(X', Y') \wedge R_i(X, X')\}_{i \in I}$$

containing expressions in which, at least in the case where $Dom(X)$ and $Dom(Y)$ are finite, all occurrences of variables X and X' can be eliminated.

Example : a reader/writer problem

We describe a simple readers/writers system by the following "*program*" RW; in fact RW defines a family of labeled transition relations where for readability reasons an explicit label ((b-read),(e-read),...) of each action is put between parentheses in front of the expression defining the transition relation.

$$
\begin{array}{llcl}
\multicolumn{4}{l}{RW = \{} \\
\text{(b-read)} & (Wr > 0) \wedge (Aw = 0) & \wedge & (Wr' = Wr - 1) \wedge (Ww' = Ww) \wedge \\
 & & & (Ar' = Ar + 1) \wedge (Aw' = Aw), \\
\text{(e-read)} & (Ar > 0) & \wedge & (Wr' = Wr + 1) \wedge (Ww' = Ww) \wedge \\
 & & & (Ar' = Ar - 1) \wedge (Aw' = Aw), \\
\text{(b-write)} & (Ww > 0) \wedge (Aw = 0) \wedge & & \\
 & (Ar = 0) & \wedge & (Wr' = Wr) \wedge (Ww' = Ww - 1) \wedge \\
 & & & (Ar' = Ar) \wedge (Aw' = Aw + 1), \\
\text{(e-write)} & (Aw > 0) & \wedge & (Wr' = Wr) \wedge (Ww' = Ww + 1) \wedge \\
 & & & (Ar' = Ar) \wedge (Aw' = Aw - 1), \\
\text{(n-wait)} & & & ((Wr' = Wr + 1) \vee (Ww' = Ww + 1)) \wedge \\
 & & & (Ar' = Ar) \wedge (Aw' = Aw) \\
\quad \} & & &
\end{array}
$$

where Wr and Ww are positive integer variables representing respectively the numbers of waiting readers and waiting writers, Ar and Aw respectively the numbers of active readers and active writers. The transition relation associated with RW has an infinite number of states as Wr and Ww can always be increased by action (*n-wait*).

We want to prove *mutual exclusion* between readers and writers. Then, the only relevant information is, whether the number of active readers and writers is positive or not. Therefore, we define an abstraction relation ρ mapping the program variables on two boolean variables b_1 and b_2 meaning respectively *there is no active reader* and *there is no active writer*, by

$$\rho((Wr, Ww, Ar, Aw),\ (b_1, b_2)) \ := \ (b_1 \equiv (Ar = 0)) \wedge (b_2 \equiv (Aw = 0)).$$

As ρ is a total function, RW_ρ is a faithful abstraction of RW via ρ. For each one of the five transition relations R_i of RW we have to compute the abstract transition relation

$$(R_i)_\rho = \exists X \ \exists X' \ . \ \rho(X, Y) \wedge \rho(X', Y') \wedge R_i(X, X')$$

For the transition relation R_1 (labeled by (b-read)) one obtains the following expression:

$$\begin{aligned}(R_1)_\rho = &\exists(Ar, Aw, Wr, Ww)\ \exists(Ar', Aw', Wr', Ww')\ .\\ &(b_1 \equiv (Ar = 0)) \wedge (b_2 \equiv (Aw = 0)) \ \wedge \ (b_1' \equiv (Ar' = 0)) \wedge (b_2' \equiv (Aw' = 0)) \ \wedge\\ &(Wr > 0) \wedge (Aw = 0) \wedge (Wr' = Wr - 1) \wedge\\ &(Ww' = Ww) \wedge (Ar' = Ar + 1) \wedge (Aw' = Aw)\\ &= b_2 \ \wedge \ \neg b_1' \ \wedge \ b_2'\end{aligned}$$

By doing a similar computation for all R_i we obtain the following family of abstract transition relations:

$$\begin{array}{lllcl} RW_\rho = \{ & \text{(b-read)} & b_2 & \wedge & \neg b_1' \wedge b_2', \\ & \text{(e-read)} & \neg b_1 & \wedge & (b_2' \equiv b_2), \\ & \text{(b-write)} & b_1 \wedge b_2 & \wedge & b_1' \wedge \neg b_2', \\ & \text{(e-write)} & \neg b_2 & \wedge & (b_1' \equiv b_1), \\ & \text{(n-wait)} & & & (b_1' \equiv b_1) \wedge (b_2' \equiv b_2) \ \ \} \end{array}$$

The finite global transition relation represented by RW_ρ is given graphically in Figure 1.

5. General results on property preservation

Now we have defined a notion of abstraction and a way to compute abstract programs. An important point is to know for which properties we can deduce from the satisfaction on the abstract system its satisfaction on the concrete system. In order to answer this question, we consider first the general problem of property preservation between two systems. If the property lattices of the two systems are related via some monotonic function $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$, then the satisfaction of some state property f is *preserved* from S_1 to S_2 via α if for any state of Q_1 satisfying f all states of Q_2 in its image by α satisfy property f. We have *strong preservation* if the converse holds also; this means intuitively that whenever a state of Q_1 does *not* satisfy f, then there exists a state in its image by α which does not satisfy f.

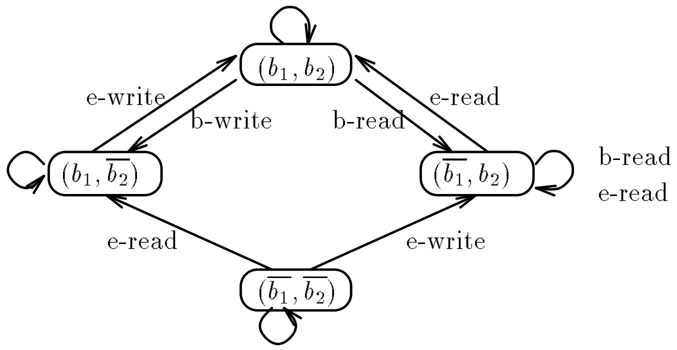

Figure 1. Readers/Writers abstraction

We give useful characterizations of these definitions if there exists a function γ such that (α, γ) is a connection, because in Section 6 we apply this notion of preservation to systems related via $\langle\alpha, \gamma\rangle$-simulation. We give also a theorem allowing to deduce strong preservation from preservation in both directions.

Let us first introduce some notations. We suppose that program properties are expressed by formulas of a logical language $\mathcal{F}(\mathcal{P})$ where $\mathcal{P} = \{P_1, P_2, ...\}$ is a set of propositional variables interpreted as sets of states. For a given system $S = (Q, R)$ and an *interpretation function* $\mathcal{I} : \mathcal{P} \rightarrow 2^Q$, the semantics of $\mathcal{F}(\mathcal{P})$ is given by means of a function $|\ |_{S,\mathcal{I}} : \mathcal{F}(\mathcal{P}) \rightarrow 2^Q$, associating with each formula its characteristic set, i. e., the set of states satisfying it. This function is such that $\forall P \in \mathcal{P} \,.\, |P|_{S,\mathcal{I}} = \mathcal{I}(P)$. To simplify notations, either one or both of the subscripts S and $\mathcal{I}$ in $|f|_{S,\mathcal{I}}$ will be omitted whenever their values can be determined by the context.

Definition 8 *(Preservation)*
Let $f \in \mathcal{F}(\mathcal{P})$ be a formula, $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems, $\Pi \subseteq Q_1$, $\mathcal{I} : \mathcal{P} \rightarrow 2^{Q_1}$ an interpretation function and $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$. We say that α preserves *(respectively* strongly preserves*) f for $\mathcal{I}$ on Π if and only if for any $q \in \Pi$,*

$$q \in |f|_{S_1,\mathcal{I}} \textit{ implies (respectively if and only if) } \alpha(\{q\}) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}.$$

If $\Pi = Q_1$, we omit to mention that the preservation is on Π.

In this definition, the function α establishes a correspondence between properties of S_1 and properties of S_2. Preservation means that the function α is compatible with the satisfaction relation. In the sequel, where the function α under consideration is always monotonic, and even such that there exists a function γ, such that (α, γ) is a connection, we use the following characterizations of the notion of preservation in order to establish preservation results.

Lemma 2 *(Characterization of preservation)*
Let $f \in \mathcal{F}(\mathcal{P})$ be a formula, $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems, $\mathcal{I} : \mathcal{P} \rightarrow 2^{Q_1}$ be an interpretation function and $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$.

1. *if α is monotonic then*
   $\alpha(|f|_{S_1,\mathcal{I}}) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}$ implies α preserves f for $\mathcal{I}$
   and if α distributes over $\cup$, the converse also holds.

2. *if there exists γ such that (α, γ) is a Galois connection, then*

   (A) α preserves f for $\mathcal{I}$ if and only if
   $|f|_{S_1,\mathcal{I}} \subseteq \gamma(|f|_{S_2,\alpha\circ\mathcal{I}})$

   (B) α strongly preserves f for $\mathcal{I}$ if and only if
   $|f|_{S_1,\mathcal{I}} \;=\; \gamma(|f|_{S_2,\alpha\circ\mathcal{I}})$

Proof: The first direction of (1) is immediate: from $q \in |f|_{S_1,\mathcal{I}}$, we obtain by monotonicity of α, $\alpha(\{q\}) \subseteq \alpha(|f|_{S_1,\mathcal{I}}) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}$. If α distributes over $\cup$, then $\alpha(|f|_{S_1,\mathcal{I}}) = \alpha(\bigcup_{q\in|f|_{S_1,\mathcal{I}}}\{q\}) = \bigcup_{q\in|f|_{S_1,\mathcal{I}}}\alpha(\{q\})$ which establishes the result.
The proof of (2A) is direct from (1) and the last item of Proposition 3. (2B) can be deduced from the fact that $(\alpha(\{q\}) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}) \;\Rightarrow\; q \in |f|_{S_1,\mathcal{I}}$ is equivalent to
$\bigcup_{\{\alpha(q)\subseteq|f|_{S_2,\alpha\circ\mathcal{I}}\}}\{q\} \subseteq |f|_{S_1,\mathcal{I}}$ and
$\bigcup_{\{\alpha(q)\subseteq|f|_{S_2,\alpha\circ\mathcal{I}}\}}\{q\} = \gamma(|f|_{S_2,\alpha\circ\mathcal{I}})$ by Proposition 5. ∎

The following theorem gives conditions under which preservation by α from S_1 to S_2 and preservation by α' from S_2 to S_1 implies strong preservation by α from S_1 to S_2. Notice that this theorem uses only the monotonicity of α and α'; the fact that there exists functions γ, γ' such that (α, γ) and (α', γ') are connections does not allow to weaken the conditions required here. Therefore, we use exactly this theorem in order to obtain the strong preservation results in the following section.

Theorem 1 *(Preservation and strong preservation)*
Let $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems. For any set $\Pi \subseteq Q_1$ and for any monotonic functions $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$ and $\alpha' : 2^{Q_2} \rightarrow 2^{Q_1}$ such that $\alpha' \circ \alpha \circ \alpha' = \alpha'$ and $Id_\Pi \subseteq \alpha' \circ \alpha$, if α preserves f for $\mathcal{I} : \mathcal{P} \rightarrow Im(\alpha')$ and α' preserves f for $\alpha \circ \mathcal{I}$ then α strongly preserves f for $\mathcal{I}$ on Π.

Proof: In order to show strong preservation by α suppose that, for $q \in \Pi$, $\alpha(\{q\}) \subseteq |f|_{S_2,\alpha\circ\,\mathcal{I}}$. We have,

$\alpha' \circ \alpha(\{q\}) \subseteq \alpha'(|f|_{S_2,\alpha\circ\mathcal{I}})$ (monotonicity of α'),
$q \in \alpha'(|f|_{S_2,\alpha\circ\mathcal{I}})$ ($Id_\Pi \subseteq \alpha' \circ \alpha$),
$q \in |f|_{S_1,\alpha'\circ\alpha\circ\mathcal{I}}$ (α' preserves f for $\alpha \circ \mathcal{I}$ and Lemma 2).

Since $\mathcal{I} : \mathcal{P} \rightarrow Im(\alpha')$, there exists an interpretation function $\mathcal{I}' : \mathcal{P} \rightarrow 2^{Q_2}$ such that $\mathcal{I} = \alpha' \circ \mathcal{I}'$. Thus $\alpha' \circ \alpha \circ \mathcal{I} = \alpha' \circ \alpha \circ \alpha' \circ \mathcal{I}' = \alpha' \circ \mathcal{I}' = \mathcal{I}$ which implies $q \in |f|_{S_1,\mathcal{I}}$.
∎

6. Preservation of the μ-calculus

Now we can tackle the problem of preservation between systems related by $\langle\alpha,\gamma\rangle$-simulation as defined in Section 3. The universe of properties that we consider is the set of properties expressible in the propositional branching-time μ-calculus [24] augmented by past time modalities, which we denote L_μ^p.

This logic subsumes in expressiveness the commonly used specification logics, such as the branching-time temporal logics CTL [9] and CTL^* [14] and also the linear-time temporal logics as PTL [34] and ETL [40].

We define fragments of the μ-calculus called L_μ, $\Box L_\mu$, $\Box L_\mu^p$, $\Diamond L_\mu$, and $\Diamond L_\mu^p$ (where p stands for logics containing past time operators). We show for two systems S_1 and S_2 that, if $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$, then α preserves $\Diamond L_\mu$ from S_1 to S_2 and $\widetilde{\gamma}$ preserves $\Box L_\mu$ from S_2 to S_1. If moreover $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$ holds, then stronger preservation results for the fragments augmented by the corresponding past time modality hold also. We obtain strong preservation of these fragments in case of simulation equivalence, i. e., existence of simulations in both directions.

In the case where the two systems are $\langle\alpha,\gamma\rangle$-bisimilar, the two functions mentioned above preserve $L_\mu^{(p)}$ and, under some conditions, they strongly preserve it.

In the first subsection, we recall the definition of the μ-calculus and its fragments and in the second subsection we give the preservation results. In the third subsection, we reformulate the verification method sketched in the introduction and apply it to the small example introduced in Section 4.2.

6.1. The propositional μ-calculus and its fragments

We recall the syntax and the semantics of the future and past propositional μ-calculus L_μ^p. Let $\mathcal{P}$ be a set of atomic propositions and $\mathcal{X}$ a set of variables. The set of the formulas of L_μ^p is defined by the following grammar:

$$f ::= \top \mid P \in \mathcal{P} \mid X \in \mathcal{X} \mid \Diamond f \mid \Diamond^{p} f \mid f \vee f \mid \neg f \mid \mu X.f$$

where f is syntactically monotonic on X, i. e., any occurrence of X in f is under an even number of negations.

As usually, the notion of free occurrences of variables in a formula is defined as in the first-order predicate calculus by considering the operator μ as a quantifier. A formula is *closed* if there are no variables occurring free in it. L_μ is the fragment in which the past operator $\Diamond^{p}$ is not allowed.

The semantics of the formulas is defined for a given transition system $S = (Q, R)$ and an interpretation function for the atomic propositions $\mathcal{I} : \mathcal{P} \rightarrow 2^Q$. A formula f with n free variables is interpreted as a function $|f|_{S,\mathcal{I}} : (2^Q)^n \rightarrow 2^Q$. In particular, a closed formula is interpreted as a set of states. The interpretation function is inductively defined as follows, for a *valuation* $V = (V_1, ..., V_n) \in (2^Q)^n$ of the variables occurring free in it.

$$
\begin{array}{lcl}
|\top|_{S,\mathcal{I}} & = & Q, \\
|P|_{S,\mathcal{I}} & = & \mathcal{I}(P), \\
|X_j|_{S,\mathcal{I}}(V) & = & V_j, \\
|f_1 \vee f_2|_{S,\mathcal{I}}(V) & = & |f_1|_{S,\mathcal{I}}(V) \cup |f_2|_{S,\mathcal{I}}(V), \\
|\neg f|_{S,\mathcal{I}}(V) & = & Q - |f|_{S,\mathcal{I}}(V), \\
|\Diamond f|_{S,\mathcal{I}}(V) & = & pre[R](|f|_{S,\mathcal{I}}(V)), \\
|\Diamond^p f|_{S,\mathcal{I}}(V) & = & post[R](|f|_{S,\mathcal{I}}(V)), \\
|\mu X.f|_{S,\mathcal{I}}(V) & = & \bigcap\{Q' \subseteq Q \ : \ |f|_{S,\mathcal{I}}[Q'/X](V) \subseteq Q'\}.
\end{array}
$$

We extend L_μ^p by adding as usually the formulas $\bot$, $f \wedge g$, $f \Rightarrow g$, $\nu X.f(X)$, $\Box f$ and $\Box^p f$ which are respectively abbreviations for $\neg\top$, $\neg(\neg f \vee \neg g)$, $\neg f \vee g$, $\neg\mu X.\neg f(\neg X)$, $\neg\Diamond\neg f$ and $\neg\Diamond^p\neg f$.

A formula of this extended language is in *positive normal form* if and only if all the negations occurring in it are applied to atomic propositions. It can be shown that any formula of L_μ^p has an equivalent formula in positive normal form.

We define fragments of L_μ^p called $\Box L_\mu$, $\Box L_\mu^p$, $\Diamond L_\mu$ and $\Diamond L_\mu^p$. Their sets of formulas are given respectively by the two following grammars where the past time modalities $\Box^p$ and $\Diamond^p$ are not allowed in the future fragments $\Box L_\mu$, respectively $\Diamond L_\mu$.

$$g ::= \top \mid \bot \mid P \mid \neg P \mid X \mid \Box g \mid \Box^p g \mid g \vee g \mid g \wedge g \mid \mu X.g \mid \nu X.g$$

$$h ::= \top \mid \bot \mid P \mid \neg P \mid X \mid \Diamond h \mid \Diamond^p h \mid h \vee h \mid h \wedge h \mid \mu X.h \mid \nu X.h$$

Notice that properties expressed by formulas of $\Box L_\mu^{(p)}$ involve only universal quantification over computation sequences (due to the use of the $\Box$ (or $\Box^p$) operator) whereas those expressed by formulas of $\Diamond L_\mu^{(p)}$ involve only existential quantification over computation sequences.

We consider the *positive* fragments $\Box L_\mu^{(p)+}$ and $\Diamond L_\mu^{(p)+}$ obtained from the above languages by forbidding the use of the negation even on atomic propositions. We consider also the fragments $L_\mu^{(p)+}$ corresponding to the subset of $L_\mu^{(p)}$ formulas in positive normal form without negations. We can translate any formula of $L_\mu^{(p)}$ which is in positive normal form into an equivalent formula in $L_\mu^{(p)+}$ by replacing negated atomic propositions, i. e., formulas in the form $\neg P$, by new atomic propositions. Thus, since any formula of $L_\mu^{(p)}$ has an equivalent formula in positive normal form, we can express in $L_\mu^{(p)+}$ any property expressible in $L_\mu^{(p)}$, modulo this encoding of the formulas $\neg P$. Obviously, the same translation can be done from $*L_\mu^{(p)}$ to $*L_\mu^{(p)+}$ for $* \in \{\Box, \Diamond\}$.

In $\Box L_\mu$ we can express branching-time properties as for instance the *safety* properties with respect to the simulation preorder [3]. The class of these properties corresponds to the fragment of $\Box L_\mu$ without the least fixpoint operator μ.

Furthermore, it can be shown that any ω-regular linear-time property, i. e., expressible by a nondeterministic Büchi automaton [6], can be expressed in $\Box L_\mu$ [4]. For example, the *safety* property *always* P can be expressed by the formula $\nu X.(P \wedge \Box X)$. Moreover, the *guarantee* property (according to [32]) *eventually* P *in any infinite computation sequence* can be expressed by the formula $\mu X.(P \vee \Box X)$. Properties in the other classes in the hierarchy given in [32] are obtained by using alternations of the μ and the ν operators. The properties of $\forall CTL^*$ can be expressed in $\Box L_\mu$ if we restrict ourselves to models whose transition relation is total as $\forall CTL^*$ allows to express general eventuality. Notice that if the transition relation of the considered models is not necessary total, "eventually P" is expressed by the formula $\mu X.(P \vee \Diamond true \wedge \Box X)$, which is neither in $\Box L_\mu$ nor in $\Diamond L_\mu$.

The formulas of $\Diamond L_\mu$ are negations of formulas of $\Box L_\mu$ and conversely.

Past time modalities can be used for two different aims: they allow to express properties which cannot be expressed using only future modalities, e. g., $\mu X.(init \vee \Box^p X)$ holds exactly in the set of states reachable from a state satisfying $init$. Moreover, they may be used in order to define alternative computation algorithms for invariants and eventually properties which in some cases converge much faster. For example, the formula $init \Rightarrow \nu X.(P \wedge \Box X)$ is equivalent to $\neg P \Rightarrow \nu X.(\neg init \wedge \Box^p X)$.

6.2. Preservation results

First, we define the notion of consistency which expresses that a chosen function relating two property lattices, $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$, preserves the meaning of the atomic propositions defined by an interpretation function $\mathcal{I}$ on 2^{Q_1}. α is consistent with $\mathcal{I}$ if for all atomic propositions the images of $\mathcal{I}(P)$ and $\overline{\mathcal{I}(P)}$ by α are disjoint, i. e., the images by α of the interpretation of P and of $\neg P$ are non contradictory. Lemma 3 says that — in the case that (α, γ) is a connection — consistency of α with $\mathcal{I}$ expresses the fact that $\widetilde{\gamma} \circ \alpha$ strongly preserves the interpretation of all atomic propositions.

Definition 9 *(Consistency)*
Let Q_1 and Q_2 be two sets of states and $\mathcal{I} : \mathcal{P} \rightarrow 2^{Q_1}$ an interpretation and $\alpha : 2^{Q_1} \rightarrow 2^{Q_2}$ any function relating the two property lattices. Then, α is consistent *with $\mathcal{I}$ if*

$$\forall P \in \mathcal{P}.\ \alpha(\overline{\mathcal{I}(P)}) \cap \alpha(\mathcal{I}(P)) = \emptyset$$

Lemma 3 *(Characterization of consistency)*
Under the same assumptions as in Definition 9, if there exists γ such that (α, γ) is a connection, then α is consistent *with $\mathcal{I}$ if and only if*

$$\forall P \in \mathcal{P}.\ \gamma(\alpha(\mathcal{I}(P))) = \mathcal{I}(P)$$

Proof: A proof by contradiction can be obtained using Proposition 7. ∎

Now, we give a theorem about the preservation in the case that two given systems S_1 and S_2 are related by $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$. The theorem says that α preserves formulas of $\Diamond L_\mu$ from S_1 to S_2, $\widetilde{\gamma}$ preserves formulas of $\Box L_\mu^{(p)}$ from S_2 to S_1 and if even $S_1 \simeq_{\langle\alpha,\gamma\rangle} S_2$ holds, then as well α as $\widetilde{\gamma}$ preserve the whole L_μ. Furthermore, if one replaces $\sqsubseteq_{\langle\alpha,\gamma\rangle}$ by $\preceq_{\langle\alpha,\gamma\rangle}$, one obtains analogous preservation results for the fragments augmented by the corresponding past modalities.

Theorem 2 *(Preservation of $\Box L_\mu^{(p)}$, $\Diamond L_\mu^{(p)}$ and $L_\mu^{(p)}$)*
Let $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ be two transition systems and $\mathcal{I}_1 : \mathcal{P} \rightarrow 2^{Q_1}$, $\mathcal{I}_2 : \mathcal{P} \rightarrow 2^{Q_2}$ two interpretation functions of atomic propositions.

1. *If $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ (respectively $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$), then*

   (A) α preserves the formulas of $\Diamond L_\mu^+$ (respectively $\Diamond L_\mu^{p+}$) for $\mathcal{I}_1$, and if α is consistent with $\mathcal{I}_1$ then α preserves the formulas of $\Diamond L_\mu$ (respectively $\Diamond L_\mu^p$) for $\mathcal{I}_1$.

   (B) $\widetilde{\gamma}$ preserves the formulas of $\Box L_\mu^+$ (respectively $\Box L_\mu^{p+}$) for $\mathcal{I}_2$, and if $\widetilde{\gamma}$ is consistent with $\mathcal{I}_2$ then $\widetilde{\gamma}$ preserves $\Box L_\mu$ (respectively $\Box L_\mu^p$) for $\mathcal{I}_2$.

2. *If $S_1 \simeq_{\langle\alpha,\gamma\rangle} S_2$ (respectively $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$ and $S_2 \preceq_{\langle\widetilde{\gamma},\widetilde{\alpha}\rangle} S_1$) then α preserves the formulas of L_μ^+ (respectively L_μ^{p+}) for $\mathcal{I}_1$ and if α is consistent with $\mathcal{I}_1$ then α preserves the formulas of L_μ (respectively L_μ^p) for $\mathcal{I}_1$.*

Proof: The proof that α preserves L_μ^+ if $S_1 \simeq_{\langle\alpha,\gamma\rangle} S_2$ consists, due to Lemma 2, in showing that for any formula $f \in L_\mu^+$ and for any valuation V, we have
$\alpha(|f|_{S_1,\mathcal{I}_1}(V)) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}_1}(\alpha(V))$.

The proof is done by induction on the structure of f, and for all operators (including fixpoint operators), except $\Diamond$ and $\Box$, we need only the monotonicity of α in order to establish this fact. For $\Diamond$ we need the fact that $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ and for $\Box$ we need the fact that $S_2 \sqsubseteq_{\langle\widetilde{\gamma},\widetilde{\alpha}\rangle} S_1$. This proof is given in Appendix A.2.

The proof of preservation of L_μ^{p+} under the condition that $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$ is obtained by Lemma 1 saying that forward and backward simulation implies $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ and ${S_1}^{-1} \sqsubseteq_{\langle\alpha,\gamma\rangle} {S_2}^{-1}$ (where $S_i = (Q_i, \mathcal{R}_i^{-1})$) and the observation that $post[\mathcal{R}] = pre[\mathcal{R}^{-1}]$.

Finally, if α is consistent with $\mathcal{I}_1$, it is straightforward to deduce that
$\alpha(|\neg P|_{S_2,\mathcal{I}_1}) \subseteq |\neg P|_{S_1,\alpha\circ\mathcal{I}_1}$.

Notice that we have also preservation of L_μ^{p+} by $\widetilde{\gamma}$ by exchanging the roles of α and $\widetilde{\gamma}$ and of S_1 and S_2 and then using symmetrical arguments. Now, the proofs of (1A) and (1B) are obvious from the fact that for the preservation of $\Diamond L_\mu^{(p)+}$ by α we need only the condition that $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ (respectively $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$), and for the preservation of $\Box L_\mu^{(p)+}$ by $\widetilde{\gamma}$ the condition that $S_1 \sqsubseteq_{\langle\widetilde{\widetilde{\alpha}},\widetilde{\widetilde{\gamma}}\rangle} S_2$ (respectively $S_1 \preceq_{\langle\widetilde{\widetilde{\alpha}},\widetilde{\widetilde{\gamma}}\rangle} S_2$), which is equivalent to $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ (respectively $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$).

■

It is known that in order to have *strong preservation* of L_μ one needs the existence of a *bisimulation* between the transition systems S_1 and S_2 (Theorem 4 gives the exact conditions). By using Theorem 1, one obtains from Theorem 2 strong preservation of fragments of L_μ under the weaker condition that is the existence of a *mutual simulation* between S_1 and S_2 and the additional conditions required in Theorem 1:

Theorem 3 *(Strong preservation of* $\Box L_\mu^{(p)}$ *and* $\Diamond L_\mu^{(p)}$*)*
Let $S_1 = (Q_1, R_1)$ *and* $S_2 = (Q_2, R_2)$ *be two transition systems. If* $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$ *and* $S_2 \sqsubseteq_{\langle\alpha',\gamma'\rangle} S_1$ *(respectively* $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$ *and* $S_2 \preceq_{\langle\alpha',\gamma'\rangle} S_1$*) for* α, α' *such that* $\alpha' \circ \alpha \circ \alpha' = \alpha'$*, then*

1. *If* $Id_\Pi \subseteq \alpha' \circ \alpha$ *for some* $\Pi \subseteq Q_1$*, then* α *strongly preserves* $\Diamond L_\mu^+$ *(respectively* $\Diamond L_\mu^{p+}$*) on* Π *for any interpretation* $\mathcal{I} : \mathcal{P} \rightarrow \Pi$*. Furthermore, if* α *is consistent with* $\mathcal{I}$*, then* α *strongly preserves* $\Diamond L_\mu$ *(respectively* $\Diamond L_\mu^p$*) for* $\mathcal{I}$ *on* Π*.*
2. *If* $Id_\Pi \subseteq \widetilde{\gamma}' \circ \widetilde{\gamma}$ *for some* $\Pi \subseteq Q_2$*, then* $\widetilde{\gamma}$ *strongly preserves* $\Box L_\mu^+$ *(respectively* $\Box L_\mu^{p+}$*) on* Π *for any interpretation* $\mathcal{I} : \mathcal{P} \rightarrow \Pi$*. Furthermore, if* $\widetilde{\gamma}$ *is consistent with* $\mathcal{I}$*, then* $\widetilde{\gamma}$ *strongly preserves* $\Box L_\mu$ *(respectively* $\Box L_\mu^p$*) for* $\mathcal{I}$ *on* Π*.*

Proof: (1) is a direct application of Theorem 1 using Theorem 2. (2) is obtained in the same way by using Proposition 8 which guarantees $\widetilde{\gamma}' \circ \widetilde{\gamma} \circ \widetilde{\gamma}' = \widetilde{\gamma}'$. ∎

Theorem 4 *(Strong preservation of* $L_\mu^{(p)}$*)*
Let $S_1 = (Q_1, R_1)$ *and* $S_2 = (Q_2, R_2)$ *be two transition systems. If* $S_1 \simeq_{\langle\alpha,\gamma\rangle} S_2$ *(respectively* $S_1 \preceq_{\langle\alpha,\gamma\rangle} S_2$ *and* $S_2 \preceq_{\langle\widetilde{\gamma},\widetilde{\alpha}\rangle} S_1$*) and* $\widetilde{\gamma} \circ \alpha \circ \widetilde{\gamma} = \widetilde{\gamma}$*, then*

1. α *strongly preserves* L_μ *(respectively* L_μ^p*) on* $Im(\widetilde{\gamma})$ *for any interpretation* $\mathcal{I}_1 : \mathcal{P} \rightarrow Im(\widetilde{\gamma})$ *and*
2. $\widetilde{\gamma}$ *strongly preserves* L_μ *(respectively* L_μ^p*) on* $Im(\alpha)$ *for any interpretation* $\mathcal{I}_1 : \mathcal{P} \rightarrow Im(\alpha)$*.*

Proof: As the preceding theorem, the proof of strong preservation by α is obtained directly from Theorems 1 and 2 by replacing α' by $\widetilde{\gamma}$ and using the fact that $Id_{Im(\widetilde{\gamma})} \subseteq \widetilde{\gamma} \circ \alpha$ (Proposition 8) and the fact that α is consistent with any $\mathcal{I}_1 : \mathcal{P} \rightarrow Im(\alpha)$ by using the same arguments as in the proof of Theorem 1. The proof for $\widetilde{\gamma}$ is symmetrical. ∎

6.3. Application

Theorem 2 provides the basis for our verification method by using abstraction. Given a program $S = (Q, R)$, a set $\mathcal{P}$ of atomic propositions occurring in formula $f \in \Box L_\mu^p$ and an interpretation function $\mathcal{I} : \mathcal{P} \rightarrow 2^Q$, one can proceed as follows in order to verify that S satisfies f, i. e., $|f|_{S,\mathcal{I}} = Q$:

(1) Give an abstraction relation $\rho \subseteq Q \times Q_A$ which is total on Q and the corresponding abstraction function $\alpha = post[\rho]$.

(2) Compute the abstract system S_ρ and verify whether the characteristic set of f on S_ρ, obtained using the interpretation function $\alpha \circ \mathcal{I}$, is contained in the image $\alpha(Q)$ of concrete states, that means we have to verify if

$\widetilde{\gamma}(|f|_{S_\rho, \alpha \circ \mathcal{I}}) = Q$.

Notice that a sufficient condition for this is that $|f|_{S_\rho, \alpha\circ\mathcal{I}} = Q_A$ expressing that f holds on S_ρ. If the answer in (2) is positive and no atomic proposition occurs negated in f, then using Theorem 2.(1B), we obtain

(3) S satisfies f with the interpretation function $\widetilde{\gamma} \circ \alpha \circ \mathcal{I}$, i. e., $|f|_{S, \widetilde{\gamma}\circ\alpha\circ\mathcal{I}} = Q$.

If furthermore, $\mathcal{I}(P) = (\widetilde{\gamma} \circ \alpha \circ \mathcal{I})(P)$ for any $P \in \mathcal{P}$ that occurs in f, then

S satisfies f under interpretation $\mathcal{I}$, i. e., $|f|_{S,\mathcal{I}} = Q$.

This means (by Lemma 3) that in order to apply the verification method one needs the consistency of α with $\mathcal{I}$ for all atomic propositions occurring non negated in f. For propositions $P \in \mathcal{P}$ occurring only negated in f, computing $|f|_{S_\rho, \alpha\circ\mathcal{I}}$ amounts to evaluate f on S with interpretation $\widetilde{\gamma}(\overline{\alpha(\mathcal{I}(P))})$ of $\neg P$; as $\widetilde{\gamma}(\overline{\alpha(\mathcal{I}(P))}) \subseteq \overline{\mathcal{I}(P)}$ is always true and as all operators in f represent monotonic functions (in f negation can only be applied to atomic propositions), we deduce that this amounts to evaluate a stronger property than f; therefore, the method can be applied even if the consistency requirement id not fulfilled for atomic propositions occurring only negated in f.

If the answer in (2) is negative, i. e., $\widetilde{\gamma}(|f|_{S_\rho, \alpha\circ\mathcal{I}}) = Q' \subset Q$, we can try to find a counter-example, showing that one of the states in $\overline{Q'}$ does not satisfy f, or we have to try with a more precise set of abstract states and corresponding connection.

Obviously, instead of the abstract system S_ρ, we can use any system S_A such that $S \sqsubseteq_{\langle\alpha,\gamma\rangle} S_A$ (respectively $S \preceq_{\langle\alpha,\gamma\rangle} S_A$ if f contains past time modalities).

A similar method is applied in [10]. The notion of homomorphism considered there corresponds to $\langle\alpha,\gamma\rangle$-*simulation* induced by relations ρ which are total functions from Q to Q_A such that α and $\widetilde{\gamma}$ are respectively consistent with the interpretation functions of the atomic propositions $\mathcal{I}$ and $\alpha \circ \mathcal{I}$. In that case, it is shown that the logic $\forall CTL^*$ is preserved from S_2 to S_1 under the condition that only infinite computation sequences are considered. This result is generalized by Theorem 2 since — under this condition — $\Box L_\mu^{(p)}$ is more expressive than $\forall CTL^*$. Furthermore, the notion of *exact homomorphism* considered there corresponds to *bisimulations* induced by relations ρ which are total functions from Q to Q_A such that α and $\widetilde{\gamma}$ are respectively consistent with the interpretation functions $\mathcal{I}$ and $\alpha \circ \mathcal{I}$. If S_1 and S_2 are related by an exact homomorphism, the logic CTL^* is strongly preserved.

This result is generalized by Theorem 2 (notice that this theorem can be applied because, if ρ is a total function, we have $\rho = \rho\rho^{-1}\rho$).
It should also be noticed that it is not important to choose a framework in which eventuality properties are preserved, as, even if they are preserved, they do in general not hold on abstract systems such as S_ρ or the abstract systems proposed in [10].

At this point, we can also discuss the choice of our notion of abstraction (S_A is a ρ-abstraction of S if and only if $S \sqsubseteq_\rho S_A$, i. e., $post[\rho] \circ pre[R] \circ \widetilde{pre}[\rho] \subseteq pre[R_A]$). Using this definition we obtain preservation of the formulas of $\Box L_\mu$ from the abstract system S_A to the concrete system S. As almost all properties we are interested in are in $\Box L_\mu$, this is a good notion of abstraction. But, as already mentioned, we can also define other notions of simulation, where the function representing a transition system is chosen to be $\widetilde{pre}[R]$, $post[R]$ or $\widetilde{post}[R]$.

Taking the first choice, we obtain preservation of the formulas of $\Diamond L_\mu$ from S_A to S. Notice that, if in Definition 4 we replace the functions pre by $\widetilde{pre}$, then S_2 is a $\langle\alpha, \gamma\rangle$-abstraction of the system S_1 under study if and only if S_2 $\langle\widetilde{\gamma}, \widetilde{\alpha}\rangle$-simulates S_1. That means that properties expressible in $\Box L_\mu$ can be verified by approximating a system by simulation from above, and properties expressible in $\Diamond L_\mu$ by approximating it by simulation from below; and similar as in [12], one may use such a pair of approximations in order to evaluate any property of L_μ. However, the reachability properties, which are the interesting properties in the fragment $\Diamond L_\mu$, do in general not hold on abstract systems, defined in a similar way as S_ρ by grouping sets of states into a single abstract state, and allowing only those abstract transitions corresponding to a transition of every corresponding concrete state. In order to obtain abstract systems allowing to verify reachability properties, we must replace the transition relation R by some transitive closure of it, such as $R_\tau^* R_{\neg\tau}$ where R_τ is the subset of transition representing "stuttering" or "non observable" steps, * denotes the transitive closure and $R_{\neg\tau}$ is the the set of "observable" or "non-stuttering" steps.

The choice to represent the transition relations R by the function $post[R]$, results in a notion of abstraction preserving only past modalities; however, as we have seen this is not very interesting, as by replacing $\sqsubseteq_\rho$ by $\preceq_\rho$ (which is not really a constraint in practice) one obtains preservation of both future and past modalities.

Reader/writer example continued

We apply the above verification method to readers-writers for which an abstraction has been calculated in Section 4.2.

Mutual exclusion between the readers and the writers can be expressed by the following formula

$$f = (Ar = 0 \wedge Aw = 0) \Rightarrow \nu X.((Ar = 0 \vee Aw = 0) \wedge \Box X).$$

This formula states that starting from a state with no active readers and writers implies that for any subsequent state, mutual exclusion holds.

Notice, that instead of propositional variables and an explicit interpretation function translating them into predicates on concrete program variables, we use these predicates on program variables directly in the formulas, which simplifies a bit the presentation in such a small example.

We have to show that ρ preserves the four basic predicates occurring in the formula, namely $(Ar = 0)$, $\neg(Ar = 0)$, $(Aw = 0)$ and $\neg(Aw = 0)$, i.e.,

$$pre[\rho](post[\rho](P)) = P$$

for each one of these predicates. This can be easily verified, e. g. for $(Ar = 0)$, we have $post[\rho](Ar = 0) = b_1$ and $pre[\rho](b_1) = (Ar = 0)$.

In order to verify f on S_ρ, we have to translate the atomic predicates by $post[\rho]$, resulting in the formula

$$f_A : \ (b_1 \wedge b_2) \Rightarrow \nu X.((b_1 \vee b_2) \wedge \Box X)$$

By using classical symbolic model checking for CTL (see e. g. in [36]), we obtain $|f_A|_{S_\rho} = true$. By Theorem 2, we have that mutual exclusion holds on the concrete program. The recent developments of BDDs [5] and tools manipulating them, allows to do this evaluation efficiently if the abstract domain is finite.

7. Compositionality of simulation with respect to parallel composition

In the previous sections we gave a method reducing the verification of a property of some program represented as a transition system $S = (Q, \mathcal{R})$ to the verification of the same property on some abstraction $S_\rho = (Q_\rho, \mathcal{R}_\rho)$.

When dealing with complex programs obtained as the parallel composition of simpler programs, the application of this method requires the computation of the corresponding *global* transition relation from which an abstraction can be computed. The question then arises whether it is possible to compute abstractions of complex programs as the parallel composition of abstractions of their components in order to avoid building the transition relation associated with the complex program. This is guaranteed if the compositionality property

$$\frac{(S_1 \sqsubseteq_{\rho_1} {S_1}') \ \text{ and } \ (S_2 \sqsubseteq_{\rho_2} {S_2}')}{(S_1 \parallel S_2) \sqsubseteq_{f(\parallel, \rho_1, \rho_2)} ({S_1}' \parallel {S_2}')}$$

holds, where $\parallel$ is a parallel composition operator and $f(\parallel, \rho_1, \rho_2)$ an abstraction relation depending on $\parallel$, ρ_1 and ρ_2.

In this section we present compositionality results for ρ-simulation for three different parallel composition operators and by taking $f(\parallel, \rho_1, \rho_2) = \rho_1 \cap \rho_2$.

There exist already many compositionality results for simulation relations with respect to parallel composition. Most of them concern synchronous composition or the particular case where the domains of the composed processes are disjoint.

Notice that an important difference with these results is that our simulations are parameterized by arbitrary relations and the relation used to obtain the abstraction of the composed system is computed from the abstraction relations applied to its components.

Another problem studied in this section is the relationship between the abstraction of the complex program and the abstraction resulting from the parallel composition of the abstractions of the components.

These results allow to compare the two approaches concerning the quality of the obtained abstractions.

7.1. Definition of parallel composition

As in Section 4.2 we consider transition systems S described by families of of transition relations represented by sets of binary predicates on a set of variables X, i.e., $S = \{R_i(X, X')\}_{i \in I}$ where the elements of I are considered as labels used for synchronization purposes in parallel composition. We use this representation of labeled transition systems as it allows us to define parallel composition of programs sharing variables.

We consider three types of parallel composition, synchronous ($\otimes$), asynchronous ($\|\!|$) and mixed ($[\!]\!|$). Mixed parallel composition is the most general one and the others can be considered as particular cases of it.

Definition 10 *(Parallel composition)*
Let $S_i = \{R_{ij}(X_i, X_i') \mid j \in I_i\}, i \in \{1,2\}$ and $A \subseteq I_1 \times I_2$ be a synchronization set (indicating which relations must synchronize). Furthermore, take $A_1 = \{i \mid \exists j.(i,j) \in A\}$ and $A_2 = \{j \mid \exists i.(i,j) \in A\}$ (A_i are the projections of A on I_1 respectively I_2). We define the operators $\|\!|$, $\otimes_A$, $[\![A]\!|$ as follows:

- *mixed composition $[\![A]\!|$:*

$$\begin{aligned} S_1 \,[\![A]\!|\, S_2 \;=\; & \{R_{1i} \wedge R_{2j} \mid (i,j) \in A\} \;\cup \\ & \{R_{1i} \wedge stable_{X_2 - X_1} \mid i \notin A_1\} \cup \{R_{2j} \wedge stable_{X_1 - X_2} \mid j \notin A_2\} \end{aligned}$$

  where for any set of variables $X = \{x_1, .., x_n\}$, $stable_X$ is the predicate $(x_1' = x_1) \wedge ... \wedge (x_n' = x_n)$.

- *synchronous composition $\otimes_A$:*
  $S_1 \otimes_A S_2 \;=\; \{R_{1i} \wedge R_{2j} \mid (i,j) \in A\}$

- *asynchronous composition $\|\!|$:*
  $S_1 \,\|\!|\, S_2 \;=\; \{R_{1i} \wedge stable_{X_2 - X_1} \mid i \in I_1\} \cup \{R_{2j} \wedge stable_{X_1 - X_2} \mid j \in I_2\}$

Comments :

- The mixed composition operator forces synchronization of pairs of transition relations belonging to A. $R_1 \llbracket A \rrbracket R_2$ can perform either moves resulting from the synchronous execution of transitions in some R_{1i} and R_{2j} such that $(i, j) \in A$, or moves performed by one component while the other remains idle. The latter corresponds to moves of either some R_{1i} for $i \notin A_1$ or of some R_{2j} for $j \notin A_2$. This operator allows to express the operators of CSP [19] or LOTOS [20] by simulating message communication by communication through common variables.
- Synchronous composition is a special case of mixed composition, where only the execution of synchronous transitions is possible. In the case where $A = I_1 \times I_2$, this operator is the same as $\wedge$, and this is the program composition operator used in TLA [26]. It can also be used to describe the parallel composition operators of SCCS [31], of S/R models [23] and the one used in [15].
- Asynchronous composition is the special case of the mixed composition where $A = \emptyset$. That means that all moves are moves of either some R_{1i} where $i \in I_1$ or of some R_{2j} where $j \in I_2$. This operator is exactly the *union* operator of UNITY [11].

Lemma 4 *Let be* $S_i = \{R_{ij}(X_i, X_i') \mid j \in I_i\}, i \in \{1, 2\}$ *and* $A \subseteq I_1 \times I_2$ *a synchronization set as before. Then,*

- $S_1 \parallel\!\mid S_2 = S_1 \llbracket \emptyset \rrbracket S_2$
- *If* A *such that* $A_1 = A_2 = \emptyset$, *then* $S_1 \otimes_A S_2 = S_1 \llbracket A \rrbracket S_2$
- *If* $A = I_1 \times I_2$, *then* $R_1 \otimes_A R_2 = R_1 \wedge R_2$
- $S_1 \llbracket A \rrbracket S_2 \; = \{R_{1i} \otimes_{\{(i,j)\}} R_{2j}\}_{(i,j) \in A} \;\parallel\!\mid\; \{R_{1i} \wedge stable_{X_2 - X_1}\}_{i \notin A_1} \;\parallel\!\mid$
  $\{R_{2j} \wedge stable_{X_1 - X_2}\}_{j \notin A_2}$
- $S_1 = \parallel\!\mid_{i \in I} \{R_{1i}\}$ *where* $\parallel\!\mid_{i \in I}$ *is the obvious n-ary extension of* $\parallel\!\mid$.

The mixed composition operator is the most general one as it allows to express the others. We prefer however to consider the three operators because they give each one rise to specific results.

The last item comes from the fact that all the R_{1i} are defined on the same set of variables.

7.2. Compositionality results

Now, we give for all operators of Definition 10, conditions on the abstraction relations ρ_i, under which the rule

$$\text{(Comp)}\quad \frac{(S_1 \sqsubseteq_{\rho_1} S_1{}') \ \text{ and } \ (S_2 \sqsubseteq_{\rho_2} S_2{}')}{S_1 \parallel S_2 \ \sqsubseteq_{\rho_1 \wedge \rho_2} S_1{}' \parallel S_2{}'}$$

holds. Furthermore, we are interested in applying this rule in the particular case where $S_i' = S_{i\rho_i}$ (defined as in Section 4.1). In that case an interesting question is whether the abstractions $R_{1_{\rho_1}} \parallel R_{2_{\rho_2}}$ and $(R_1 \parallel R_2)_{\rho_1 \wedge \rho_2}$ are comparable in order to know which way of computing abstractions gives better approximations: direct computation from the compound system or computation by composition of abstractions of the components. Intuitively one would think that

$$(R_1 \parallel R_2)_{\rho_1 \wedge \rho_2} \quad \Rightarrow \quad R_{1_{\rho_1}} \parallel R_{2_{\rho_2}}$$

holds always. However, the second items of the following theorems show that this is only true without restrictions for synchronous parallel composition. Notice also that, if this implication holds, then also the rule, obtained by replacing in (Comp) the simulation preorder $\sqsubseteq$ by forward and backward simulation $\preceq$. In fact, in order to obtain this modified rule, slightly weaker conditions than those required for the second item of the following theorems are necessary, but as even the stronger ones are almost always satisfied in practice, we propose the interested reader to look at [27] for more details.

The third items of the following theorems show that in order to obtain the inverse implication

$$R_{1_{\rho_1}} \parallel R_{2_{\rho_2}} \quad \Rightarrow \quad (R_1 \parallel R_2)_{\rho_1 \wedge \rho_2}$$

for synchronous composition relatively strong conditions are necessary; whereas for asynchronous composition the conditions are relatively easy to fulfil.

Assumption 1 *Throughout the rest of the section we consider a set of variables X of the form $X_1 \cup X_2$ where X_1 and X_2 are not necessarily disjoint, two transition systems $S_i = \{R_{ij}(X_i, X_i') \mid j \in I_i\}, i \in \{1,2\}$ and $X_A = X_{1A} \cup X_{2A}$ a set of abstract variables.*

We denote also $X_c = X_1 \cap X_2$, the set of common variables, $X_{il} = X_i - X_c$ the local variables of S_i and analogously $X_{cA} = X_{1A} \cap X_{2A}$, the set of common abstract variables and $X_{ilA} = X_{iA} - X_{cA}$ the local abstract variables of S_i.

We consider also two relations relating the concrete and the abstract domains, $\rho_i(X_i, X_{iA})$, which are total on X_i and such that $\rho_1 \wedge \rho_2$ is total on X. In order to simplify the expression of the results and because it does not restrict generality, we suppose in the sequel that the relations ρ_i can be put into the form $\rho_i = \rho_{il}(X_i, X_{ilA}) \wedge \rho_{ic}(X_i, X_{cA})$, i. e., the abstract local and common variables do not depend on each other. This implies that the totality of $\rho_1 \wedge \rho_2$ is equivalent to the totality of ρ_{1l}, ρ_{2l} and $\rho_{1c} \wedge \rho_{2c}$.

Theorem 5 *(compositionality with respect to $\otimes$)*
Under the hypotheses of Assumption 1, one has

1. *If* $\rho_{ic} : X_c \rightarrow X_{cA}$ *are functions for* $i = \{1,2\}$*, then*

$$\frac{S_i \ \sqsubseteq_{\rho_i} \ S'_i, \ i = 1,2}{S_1 \otimes_A S_2 \ \sqsubseteq_{\rho_1 \wedge \rho_2} S'_1 \otimes_A S'_2}$$

2. $(R_1 \otimes_A R_2)_{\rho_1 \wedge \rho_2} \ \Rightarrow \ R_{1_{\rho_1}} \otimes_A R_{2_{\rho_2}}$

3. *If* $\rho_{ic} : X_{cA} \rightarrow X_c$ *are functions for* $i = \{1,2\}$*, then*
$R_{1_{\rho_1}} \otimes_A R_{2_{\rho_2}} \ = \ (R_1 \otimes_A R_2)_{\rho_1 \wedge \rho_2}$

Proof: The proof is rather technical — except that of (2) which uses only monotonicity arguments — and is deferred to Appendix A.3. ■

Theorem 6 *(compositionality with respect to* $\|$*)*
Under the hypotheses of Assumption 1 and if furthermore $\rho_{il} = \rho_{il}(X_{il}, X_{ilA})$*, we get*

1. *if* $\rho_{ic} = \rho_c(X_c, X_{cA})$*, then*

$$\frac{S_i \ \sqsubseteq_{\rho_i} \ S'_i, \ i = 1,2}{S_1 \| S_2 \ \sqsubseteq_{\rho_1 \wedge \rho_2} S'_1 \| S'_2}$$

2. *If* $\rho_{il} : X_{il} \rightarrow X_{ilA}$ *are functions for* $i = \{1,2\}$*, then*
$(R_1 \| R_2)_{\rho_1 \wedge \rho_2} \ \Rightarrow \ R_{1_{\rho_1}} \| R_{2_{\rho_2}}$

3. *If* $\rho_{ic} = \rho_c(X_c, X_{cA})$ *and* ρ_{il} *are onto* $(\forall X_{ilA} \ \exists X_{il}.\rho_{il}(X_{il}, X_{ilA}))$*, then*
$R_{1_{\rho_1}} \| R_{2_{\rho_2}} \ \Rightarrow \ (R_1 \| R_2)_{\rho_1 \wedge \rho_2}$

Proof: The complete proof is given in Appendix A.3. ■

Theorem 7 *(compositionality with respect to* $[\![\|$*)*
Under the same hypotheses as in Theorem 6, we get

1. *If* $\rho_{ic} : X_c \rightarrow X_{cA}$ *are functions for* $i = \{1,2\}$*, then*

$$\frac{S_i \ \sqsubseteq_{\rho_i} \ S'_i, \ i = 1,2}{S_1 \, [\![A]\!| \, S_2 \ \sqsubseteq_{\rho_1 \wedge \rho_2} S'_1 \, [\![A]\!| \, S'_2}$$

2. *If* $\rho_{il} : X_{il} \rightarrow X_{ilA}$ *are functions for* $i = \{1,2\}$*, then*
$(R_1 \, [\![A]\!| \, R_2)_{\rho_1 \wedge \rho_2} \ \Rightarrow \ R_{1_{\rho_1}} \, [\![A]\!| \, R_{2_{\rho_2}}$

3. *If* $\rho_{ic} : X_{cA} \rightarrow X_c$ *are functions for* $i = \{1,2\}$ *and* ρ_{il} *are onto, then*
$R_{1_{\rho_1}} \, [\![A]\!| \, R_{2_{\rho_2}} \ \Rightarrow \ (R_1 \, [\![A]\!| \, R_2)_{\rho_1 \wedge \rho_2}$

Proof: The fact that $R_1 \, [\![\| \, R_2$ can be expressed by using only $\otimes$ and $\|$ as given in Lemma 4, and that the conditions of both of the preceding theorems are satisfied in each of the corresponding points is enough to prove the theorem. ■

8. Conclusion

The paper studies property preserving transformations for reactive systems. A key idea is the use of $\langle\alpha,\gamma\rangle$-simulation which is the same as the standard simulation (parameterized by a relation ρ) often used to define implementations. Furthermore, $\langle\alpha,\gamma\rangle$-simulations induce abstract interpretations and this allows to apply an existing powerful theory for program analysis.

The results presented can be adapted so as to be applied to preorders and equivalences that are defined in terms of simulations or bisimulations with silent actions. For instance, one can define a $\langle\alpha,\gamma\rangle$-observational equivalence by considering as models, labeled transition systems with silent actions and using the well-known fact that observational equivalence is strong bisimulation equivalence on a modified transition relation.

An important issue is the application of the results to the verification of non-trivial systems. For this, a key problem is the choice of appropriate abstraction relations depending on the properties to be verified. In general, this task requires a deep knowledge of the concrete program to be verified and cannot be automated. However, the predicates occurring in the formula and the requirements for the preservation of these predicates help finding the minimal necessary abstract domain. Also the results of Section 7 are helpful for the user of the method as appropriate abstractions for components are easier to find than abstraction for the compound system.

In the case that both, the concrete and the abstract domains are finite, once an abstraction relation is given, the rest of the method can be mechanized: computation of the abstraction, verification of the formula and checking preservation of the predicates. We have implemented a symbolic verification tool supporting this method for finite state programs encoded as BDDs [17], [27]: Programs are parallel compositions of components which are predicates (just as the program used in the example in Section 4.2) on boolean variables. An abstract program may be obtained by composing and abstracting the components in any order using abstraction relations ρ_i given by predicates on abstract and concrete variables. Internally all predicates are represented by BDDs. A symbolic model checker allows the verification of properties. Using this tool, we have verified a protocol described in [13]. For this protocol, the use of the compositionality results of Section 7 was essential in order to be able to compute an appropriate abstract system.
In [18], we applied the same verification method to an infinite state system, a distributed cache memory [13] which is known to be difficult to verify. For this example, the abstract program could not be obtained fully automatically. It has been computed from the concrete program by replacing every concrete basic operation (operation on integers, memories and buffers) by a corresponding abstract operation on very reduced abstract domains. This example shows that our results can be

applied for the computation of abstractions of infinite state systems. However, the computation of finite abstractions of infinite state systems deserve further study.

Appendix A

A.1. Proof of Proposition 11

We want to show that if $S = (Q, R)$ is a transition system and and $\rho \subseteq Q \times Q_A$ total on Q such that $\rho = \rho\rho^{-1}\rho$, then S_ρ is a faithful abstraction of S via ρ. More precisely, we want to show that for any $S_A = (Q_A, R_A)$ such that $S \sqsubseteq_\rho S_A$ and $R_A \subseteq R_\rho$, S_A and S_ρ are $\rho^{-1}\rho$-bisimilar.

- First, let us show that $S_\rho \sqsubseteq_{\rho^{-1}\rho} S_A$, that is,

  $$post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] \subseteq pre[R_A]. \quad (*)$$

  By definition, we have $pre[R_\rho] = post[\rho] \circ pre[R] \circ pre[\rho]$. Thus, by substitution we obtain
  $post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] = post[\rho^{-1}\rho] \circ post[\rho] \circ pre[R] \circ pre[\rho] \circ \widetilde{pre}[\rho^{-1}\rho]$.
  By Proposition 2 and by the fact that $pre[\rho] = post[\rho^{-1}]$, we obtain
  $post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] = post[\rho\rho^{-1}\rho] \circ pre[R] \circ post[\rho^{-1}] \circ \widetilde{pre}[\rho^{-1}] \circ \widetilde{pre}[\rho]$.
  Now, since $\rho = \rho\rho^{-1}\rho$ (by hypothesis) and $(post[\rho^{-1}], \widetilde{pre}[\rho^{-1}])$ is a connection (by Proposition 6), we have, $post[\rho\rho^{-1}\rho] = post[\rho]$ and
  $post[\rho^{-1}] \circ \widetilde{pre}[\rho^{-1}] \subseteq Id$, thus,
  $post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] \subseteq post[\rho] \circ pre[R] \circ \widetilde{pre}[\rho]$.
  Finally, since by hypothesis we have,
  $S \sqsubseteq_\rho S_A$, i.e., $post[\rho] \circ pre[R] \circ \widetilde{pre}[\rho] \subseteq pre[R_A]$, we obtain,
  $post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] \subseteq pre[R_A]$ which is (*).

- Now, since $(\rho^{-1}\rho)^{-1} = (\rho^{-1}\rho)$, we show that $S_A \sqsubseteq_{\rho^{-1}\rho} S_\rho$, that is to say,
  $post[\rho^{-1}\rho] \circ pre[R_A] \circ \widetilde{pre}[\rho^{-1}\rho] \subseteq pre[R_\rho]$.
  By hypothesis, we have $R_A \subseteq R_\rho$, such it is sufficient to show that

  $$post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] \subseteq pre[R_\rho] \quad (**)$$

  As in the first part of the proof we obtain,
  $post[\rho^{-1}\rho] \circ pre[R_\rho] \circ \widetilde{pre}[\rho^{-1}\rho] \subseteq post[\rho] \circ pre[R] \circ pre[\rho]$,
  which is equivalent to (**). ■

A.2. Proof of Theorem 2

In order to complete the proof of Theorem 2 it remains to show that if $S_1 = (Q_1, R_1)$ and $S_2 = (Q_2, R_2)$ are transition systems and $\mathcal{I} : \mathcal{P} \rightarrow 2^{Q_1}$ is an interpretation function, such that $S_1 \simeq_{\langle\alpha,\gamma\rangle} S_2$ then α preserves the formulas of L_μ^+ for $\mathcal{I}$. By Lemma 2, it is sufficient to prove that for any formula f in L_μ^+ and for any valuation V, we have

$$\alpha(|f|_{S_1,\mathcal{I}}(V)) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}(\alpha(V)) \text{ or equivalently } |f|_{S_1,\mathcal{I}}(V) \subseteq \gamma(|f|_{S_2,\alpha\circ\mathcal{I}}(\alpha(V))).$$

To simplify the notations, we omit the valuation V whenever it is not relevant in a proof.

- $\alpha(|\bot|_{S_1,\mathcal{I}}) \subseteq |\bot|_{S_2,\alpha\circ\mathcal{I}}$ and $\alpha(|\top|_{S_1,\mathcal{I}}) \subseteq |\top|_{S_2,\alpha\circ\mathcal{I}}$ as $\alpha(\emptyset) = \emptyset$ and $\alpha(Q_1) \subseteq Q_2$.
- $\alpha(|P|_{S_1,\mathcal{I}}) = |P|_{S_2,\alpha\circ\mathcal{I}}$ by definition of the interpretation function.
- $\alpha(|X_j|_{S_1,\mathcal{I}}(V)) = \alpha(V_j) = |X_j|_{S_2,\alpha\circ\mathcal{I}}(\alpha(V))$
- $|\Box f|_{S_1,\mathcal{I}} = \widetilde{pre}[R_2](|f|_{S_1,\mathcal{I}})$ by definition of the semantics. The dual of the condition for $S_1 \sqsubseteq_{\langle\widetilde{\gamma},\widetilde{\alpha}\rangle} S_2$ is $\widetilde{pre}[R_1] \subseteq \gamma \circ \widetilde{pre}[R_2] \circ \alpha$. By substitution, we get,

  $$|\Box f|_{S_1,\mathcal{I}} \subseteq \gamma \circ \widetilde{pre}[R_2] \circ \alpha(|f|_{S_1,\mathcal{I}}).$$

  By induction hypothesis — $\alpha(|f|_{S_1,\mathcal{I}}) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}$ — we obtain,

  $$|\Box f|_{S_1,\mathcal{I}} \subseteq \gamma \circ \widetilde{pre}[R_2](|f|_{S_2,\alpha\circ\mathcal{I}}),$$

  which is equivalent to

  $$|\Box f|_{S_1,\mathcal{I}} \subseteq \gamma(|\Box f|_{S_2,\alpha\circ\mathcal{I}}).$$

- $\alpha(|\Diamond f|_{S_1,\mathcal{I}}) = \alpha \circ pre[R_1](|f|_{S_1,\mathcal{I}})$
  by definition of the semantics and monotonicity of α. As $Id_{Q_1} \subseteq \gamma \circ \alpha$, we get

  $$\alpha(|\Diamond f|_{S_1,\mathcal{I}}) \subseteq \alpha \circ pre[R_1] \circ \gamma \circ \alpha(|f|_{S_1,\mathcal{I}}).$$

  As $S_1 \sqsubseteq_{\langle\alpha,\gamma\rangle} S_2$, i. e., $\alpha \circ pre[R_1] \circ \gamma \subseteq pre[R_2]$, we get

  $$\alpha(|\Diamond f|_{S_1,\mathcal{I}}) \subseteq pre[R_2] \circ \alpha(|f|_{S_1,\mathcal{I}}).$$

  By induction hypothesis — $\alpha(|f|_{S_1,\mathcal{I}}) \subseteq |f|_{S_2,\alpha\circ\mathcal{I}}$ — follows

  $$\alpha(|\Diamond f|_{S_1,\mathcal{I}}) \subseteq pre[R_2](|f|_{S_2,\alpha\circ\mathcal{I}}) = |\Diamond f|_{S_2,\alpha\circ\mathcal{I}}.$$

- $\alpha(|f_2 \vee f_1|_{S_1,\mathcal{I}}) = \alpha(|f_2|_{S_1,\mathcal{I}} \cup |f_1|_{S_1,\mathcal{I}})$
  by definition of the interpretation function. As α distributes over $\cup$, we have

  $$\alpha(|f_2 \vee f_1|_{S_1,\mathcal{I}}) = \alpha(|f_2|_{S_1,\mathcal{I}}) \cup \alpha(|f_1|_{S_1,\mathcal{I}}).$$

  By induction hypothesis, we obtain

  $$\alpha(|f_2 \vee f_1|_{S_1,\mathcal{I}}) \subseteq |f_2|_{S_2,\alpha\circ\mathcal{I}} \cup |f_1|_{S_2,\alpha\circ\mathcal{I}} = |f_2 \vee f_1|_{S_2,\alpha\circ\mathcal{I}}.$$

- An analogous proof can be obtained for conjunction.
- $|\mu X.f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}(\alpha(V)) = \bigcap\{P_2\subseteq Q_2 \ : \ |f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}[P_2/X](\alpha(V)) \subseteq P_2\}$, where V is a valuation on Q_1 of the free variables of f. As (α,γ) is a connection $(\alpha\circ\gamma \subseteq Id_{Q_2})$ and γ is monotonic,

  $$|f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}[P_2/X](\alpha(V)) \subseteq P_2 \ (*)$$

  implies

  $$\gamma(|f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}[\alpha(\gamma(P_2))/X](\alpha(V))) \subseteq \gamma(P_2).$$

  Using the induction hypothesis for f with valuation $\gamma(P_2)$ for X gives

  $$|f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(|f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}[\alpha(\gamma(P_2))/X](\alpha(V)))$$

  which implies finally by transitivity,

  $$|f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(P_2) \ \ (**).$$

  Thus, every P_2 satisfying (*) satisfies also (**). This implies $\bigcap\{P_2 : (**)\} \subseteq \bigcap\{P_2 : (*)\}$, i. e.,

  $$\begin{array}{l}\bigcap\{P_2 \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(P_2)\} \ \subseteq \\ \bigcap\{P_2 \ : \ |f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}[P_2/X](\alpha(V)) \subseteq P_2\} = |\mu X.f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}(\alpha(V)).\end{array}$$

  By distributivity of γ over intersection, we obtain

  $$\bigcap\{\gamma(P_2) \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(P_2)\} \subseteq \gamma(|\mu X.f|_{\mathcal{S}_2,\alpha\circ\mathcal{I}}(\alpha(V)) \ ).$$

  It remains to show that $\{\gamma(P_2) \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(P_2)\}$ contains the least fixpoint $|\mu X.f|_{\mathcal{S}_1,\mathcal{I}}(V)$. From the fact that,

  $$\{\gamma(P_2) \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(P_2)\} \subseteq \{P_1\subseteq Q_1 \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[P_1/X](V) \subseteq P_1\}$$

  we deduce that

  $$\begin{array}{l}\bigcap\{\gamma(P_2) \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[\gamma(P_2)/X](V) \subseteq \gamma(P_2)\} \ \supseteq \\ \bigcap\{P_1\subseteq Q_1 \ : \ |f|_{\mathcal{S}_1,\mathcal{I}}[P_1/X](V) \subseteq P_1\} = |\mu X.f|_{\mathcal{S}_1,\mathcal{I}}(V)\end{array}$$

  which completes the proof.
- An analogous proof can be obtained for the greatest fixpoint. ■

A.3. Proofs of Theorems 5 and 6

We suppose all the notations and hypotheses introduced in Assumption 1 of Section 7.2 for the formulation of the two theorems. Furthermore, we use the following notation for the composition of relations:

If $R_1(X,X')$ and $R_2(X',X'')$ are predicates representing relations, we represent by $R_1 * R_2$ the composition of the relations, i. e., $R_1 * R_2$ represents the predicate $\exists X' \ . \ R_1(X,X') \wedge R_2(X',X'')$.

A.3.1. Proof of Theorem 5

1. In order to show the stability of $\sqsubseteq$ with respect to synchronous composition $\otimes$ we use Definition 5. More precisely, we show that,

   (*) ($\bigvee_{i\in I_1} R_{1i})^{-1} * \rho_1 \;\Rightarrow\; \rho_1 * R_1'^{-1}$ and ($\bigvee_{j\in I_2} R_{2j})^{-1} * \rho_2 \;\Rightarrow\; \rho_2 * R_2'^{-1}$
   implies
   (**) ($\bigvee_{(i,j)\in A}(R_{1i} \wedge R_{2j})^{-1}) * (\rho_1 \wedge \rho_2) \;\Rightarrow\; (\rho_1 \wedge \rho_2) * (R_1' \wedge R_2')^{-1}$.

   where (*) can be expressed as:
   $\forall(X'_{1l}, X'_c)\forall(X_{1lA}, X_{cA})$.
   $(\exists(X_{1l}, X_c) \,.\, \rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA})) \wedge \exists i \,.\, R_{1i}((X_{1l}, X_c), (X'_{1l}, X'_c)) \;\Rightarrow$
   $\exists(X'_{1lA}, X'_{cA}) \,.\, \rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA})) \wedge R'_1((X_{1lA}, X_{cA}), (X'_{1lA}, X'_{cA}))\ )$
   $\wedge$
   $\forall(X'_c, X'_{2l})\forall(X_{cA}, X_{2lA})$.
   $(\exists(X_c, X_{2l}) \,.\, \rho_2((X_c, X_{2l}), (X_{cA}, X_{2lA})) \wedge \exists j \,.\, R_{2j}((X_c, X_{2l}), (X'_c, X'_{2l})) \;\Rightarrow$
   $\exists(X'_{cA}, X'_{2lA}) \,.\, \rho_2((X'_c, X'_{2l}), (X'_{cA}, X'_{2lA})) \wedge R'_2((X_{cA}, X_{2lA}), (X'_{cA}, X'_{2lA}))\ )$.

   and (**) can be expressed as:
   $\forall(X'_{1l}, X'_c, X'_{2l})\forall(X_{1lA}, X_{cA}, X_{2lA})$.
   $(\exists(X_{1l}, X_c, X_{2l}) \,.\, \rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA})) \wedge \rho_2((X_c, X_{2l}), (X_{cA}, X_{2lA})) \wedge$
   $\exists(i, j) \in A \,.\, R_{1i}((X_{1l}, X_c), (X'_{1l}, X'_c)) \wedge R_{2j}((X_c, X_{2l}), (X'_c, X'_{2l}))$
   $\Rightarrow$
   $\exists(X'_{1lA}, X'_{cA}, X'_{2lA}) \,.\, \rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA})) \wedge \rho_2((X'_c, X'_{2l}), (X'_{cA}, X'_{2lA})) \wedge$
   $R'_1((X_{1lA}, X_{cA}), (X'_{1lA}, X'_{cA})) \wedge R'_2((X_{cA}, X_{2lA}), (X'_{cA}, X'_{2lA}))\ )$.

   It is quite easy to see that if we choose the same X'_c and X_{cA} in part 1 and 2 of (*), and if we can choose the same X_c, then totality of $\rho_1 \wedge \rho_2$ on $Dom(X)$ is sufficient to be able to choose the same X'_{cA} such that both $\rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA}))$ and $\rho_2((X'_c, X'_{2l}), (X'_{cA}, X'_{2lA}))$.

   The fact that that ρ_{ic} are (the same) functions guaranties that if there exists a X'_{cA} that can be chosen then it is unique, which induces by (*) that $R'_1((X_{1lA}, X_{cA}), (X'_{1lA}, X'_{cA}))$ and $R'_2((X_{cA}, X_{2lA}), (X'_{cA}, X'_{2lA}))$. This implies (**). Notice that the required conditions on ρ_i are also necessary if no more information on the transition relations R_i and R'_i is available.

2. We have to show that $(R_1 \otimes_A R_2)_{\rho_1\wedge\rho_2} \;\Rightarrow\; R_{1\rho_1} \otimes_A R_{2\rho_2}$.
   $(R_1 \otimes_A R_2)_{\rho_1\wedge\rho_2} = \bigvee_{(i,j)\in A} (\rho_1 \wedge \rho_2)^{-1} * (R_{1i} \wedge R_{2j}) * (\rho_1 \wedge \rho_2)$.
   As $\rho_1 \wedge \rho_2 \;\Rightarrow\; \rho_i$, $R_{1i} \wedge R_{2j} \;\Rightarrow\; R_{1i}$ and $R_{1i} \wedge R_{2j} \;\Rightarrow\; R_{2j}$, we have
   $\bigvee_{(i,j)\in A} (\rho_1 \wedge \rho_2)^{-1} * (R_{1i} \wedge R_{2j}) * (\rho_1 \wedge \rho_2) \;\Rightarrow$
   $\bigvee_{(i,j)\in A} (\rho_1^{-1} * R_{1i} * \rho_1) \;\wedge\; (\rho_2^{-1} * R_{2j} * \rho_2)$ which is equivalent to $R_{1\rho_1} \otimes_A R_{2\rho_2}$.

3. We have to show the inverse implication of (2). We show that
$\forall (i,j) \in A \ . \ R_{1i\rho_1} \wedge R_{2j\rho_2} \ \Rightarrow \ (R_{1i} \wedge R_{2j})_{\rho_1 \wedge \rho_2}$:

$$R_{1i\rho_1} \wedge R_{2j\rho_2}((X_{1lA}, X_{cA}, X_{2lA}), (X'_{1lA}, X'_{cA}, X'_{2lA})) \ = \\ \exists (X_{1l}, X_c) \ \exists (X'_{1l}, X'_c) \ . \\ \rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA})) \wedge R_{1i}((X_{1l}, X_c), (X'_{1l}, X'_c)) \wedge \rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA})) \\ \underline{\wedge} \\ \exists (X_c, X_{2l} \exists (X'_c, X'_{2l}) \ . \\ \rho_2((X_c, X_{2l}), (X_{cA}, X_{2lA})) \wedge R_{2j}((X_c, X_{2l}), (X'_c, X'_{2l})) \wedge \rho_2((X'_c, X'_{2l}), (X'_{cA}, X'_{2lA}))$$

The expression for $(R_{1i} \wedge R_{2j})_{\rho_1 \wedge \rho_2}$ differs from this one by the fact all the existential quantifications have to be put outside of the main conjunction (underlined), i. e., in both subexpressions the same X_c and X'_c must be chosen (this is a different proof of implication (2)). In order to get the implication (3), we must be sure that choosing in both existential quantifications the same X_c and the same X'_c we do not obtain less transitions than without this constraint. This is obviously guaranteed by the condition that ρ_{ic} are functions from X_{cA} into X_c. Notice that the required condition is also necessary if no more information on the transition relations R_i and R'_i is available. ■

A.3.2. Proof of Theorem 6

1. In order to show the stability of $\sqsubseteq$ with respect to $\parallel\!\mid$, we use again Definition 5; so we show that

$(*)$ $(\bigvee_{i \in I_1} R_{1i})^{-1} * \rho_1 \ \Rightarrow \ \rho_1 * R_1'^{-1}$ and $(\bigvee_{j \in I_2} R_{2j})^{-1} * \rho_2 \ \Rightarrow \ \rho_2 * R_2'^{-1}$
implies
$(***)$ $((\bigvee_{i \in I_1} R_{1i})) \parallel\!\mid (\bigvee_{j \in I_2} R_{2j})))^{-1} * (\rho_1 \wedge \rho_2) \ \Rightarrow \ (\rho_1 \wedge \rho_2) * (R'_1 \parallel\!\mid R'_2)^{-1}$.

As composition of relations distributes over disjunction, it is sufficient to show that
$\forall i \in I_1 \ . \ (R_{1i}^{-1} \wedge stable_{X_{2l}}) * (\rho_1 \wedge \rho_2) \ \ \Rightarrow \ (\rho_1 \wedge \rho_2) * (R_1'^{-1} \wedge stable_{X_{2lA}})$ and analogously for R_2. We show the implication for some R_{1i}.

$(R_{1i}^{-1} \wedge stable_{X_{2l}}) * (\rho_1 \wedge \rho_2) \ \Rightarrow \ (\rho_1 \wedge \rho_2) * (R_1'^{-1} \wedge stable_{X_{2lA}})$ can be expressed as

$$\forall (X'_{1l}, X'_c, \underline{X'_{2l}}) \ \forall (X_{1lA}, X_{cA}, \underline{X_{2lA}}) \ . \\ (\ \exists (X_{1l}, X_c, \underline{X_{2l}}) \ . \ \rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA})) \ \wedge \ \underline{\rho_2((X_c, X_{2l}), (X_{cA}, X_{2lA}))} \ \wedge \\ R_{1i}((X_{1l}, X_c), (X'_{1l}, X'_c)) \ \wedge \ \underline{X_{2l} = X'_{2l}} \\ \Rightarrow \\ \exists (X'_{1lA}, X'_{cA}, \underline{X'_{2lA}}) \ . \ \rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA})) \ \wedge \ \underline{\rho_2((X'_c, X'_{2l}), (X'_{cA}, X'_{2lA}))} \ \wedge \\ R_1((X_{1lA}, X_{cA}), (X'_{1lA}, X'_{cA})) \ \wedge \ \underline{X_{2lA} = X'_{2lA}} \ ).$$

This expression differs from the first conjunct of (*) (see its expression in the previous proof item (1)) by adding all the *underlined* parts. Thus, it is sufficient to show that $\rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA}))$ and $\rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA}))$ and $\rho_2((X_c, X_{2l}), (X_{cA}, X_{2lA}))$ implies $\rho_2((X'_c, X_{2l}), (X'_{cA}, X_{2lA}))$.

This is guaranteed by the the fact that ρ_{2l} does not depend on X_c and the fact that ρ_{2c} coincides with ρ_{1c}.

2. We show that
$\forall i \in I_1 \,.\, (\rho_1 \wedge \rho_2)^{-1} * (R_{1i} \wedge stable_{X_{2l}}) * (\rho_1 \wedge \rho_2) \;\Rightarrow\; ({\rho_1}^{-1} * R_{1i} * \rho_1) \wedge stable_{X_{2lA}}$
and analogously for R_2. We have,

$$\begin{array}{l}
E_1 = \\
((\rho_1 \wedge \rho_2)^{-1} * (R_{1i} \wedge stable_{X_{2l}}) * (\rho_1 \wedge \rho_2))((X_{1lA}, X_{cA}, X_{2lA}), (X'_{1lA}, X'_{cA}, X'_{2lA})) = \\
\exists (X_{1l}, X_c, \underline{X_{2l}})\ \exists (X'_{1l}, X'_c, \underline{X'_{2l}}) \,.\, \rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA})) \;\wedge \\
\underline{\rho_2((X_c, X_{2l}), (X_{cA}, X_{2lA}))} \;\wedge \\
R_1((X_{1l}, X_c), (X'_{1l}, X'_c)) \;\wedge\; \rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA})) \;\wedge \\
\underline{\rho_2((X'_c, X'_{2l}), (X'_{cA}, X'_{2lA}))} \;\wedge\; \underline{X_{2l} = X'_{2l}}
\end{array}$$

whereas

$$\begin{array}{l}
E_2 = \\
({\rho_1}^{-1} * R_1 * \rho_1 \wedge stable_{X_{2lA}})((X_{1lA}, X_{cA}, X_{2lA}), (X'_{1lA}, X'_{cA}, X'_{2lA})) \;= \\
\exists (X_{1l}, X_c)\ \exists (X'_{1l}, X'_c) \,.\, \rho_1((X_{1l}, X_c), (X_{1lA}, X_{cA})) \wedge R_1((X_{1l}, X_c), (X'_{1l}, X'_c)) \;\wedge \\
\rho_1((X'_{1l}, X'_c), (X'_{1lA}, X'_{cA})) \;\wedge\; \underline{X_{2lA} = X'_{2lA}}
\end{array}$$

where the underlining indicates the differences between the two expressions. In order to obtain $E_1 \Rightarrow E_2$ it is sufficient to show that $E_1 \;\Rightarrow\; (X_{2lA} = X'_{2lA})$. This is guaranteed by the condition that ρ_{2l} is a function, i. e., that for any X_{2l} there exists a unique X_{2lA} such that $\rho_{2l}(X_{2l}, X_{2lA})$.

3. In order to obtain (3), i. e., $E_2 \Rightarrow E_1$, it is sufficient to show that
$E_2 \;\Rightarrow\; \exists X_{2l} \,.\, \rho_{2l}(X_{2l}, X_{2lA}) \wedge \rho_{2c}(X_c, X_{cA}) \wedge \rho_{2c}(X'_c, X'_{cA})$ which is guaranteed by the fact that ρ_{2l} is onto and that ρ_{2c} coincides with ρ_{1c}. ∎

References

1. M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2), 1991. first published as Report SRC-29, DEC Research Center in 1988.
2. A. Bouajjani, S. Bensalem, C. Loiseaux, and J. Sifakis. Property preserving simulations. In *Workshop on Computer-Aided Verification (CAV), Montréal*. LNCS 630, June 1992.
3. A. Bouajjani, J.-C. Fernandez, S. Graf, J. Sifakis, and C. Rodriguez. Safety for branching semantics. In *18th ICALP, Madrid*. LNCS 510, Springer Verlag, 1991.

4. A. Bouajjani. From Linear-Time Propositional Temporal Logics to a Branching-Time μ-calculus. RTC 15, LGI-IMAG, Grenoble, 1989.

5. R. E. Bryant. Graph based algorithms for boolean function manipulation. *IEEE Trans. on Computation*, 35 (8), 1986.

6. J. R. Büchi. On a decision method in restricted second order arithmetic. In *International Congress on Logic, Method and Philosophical Science*. Stanford University Press, 1962.

7. P. Cousot and R. Cousot. Systematic design of program analysis framework. In *Proc. 6th ACM Symp. on Principle of Programming Languages*, 1979.

8. P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In *Conference on Programming Language Implementation and Logic Programming, PLILP'92*. LNCS 631, Springer Verlag, 1990.

9. E.M. Clarke, E.A. Emerson, and E. Sistla. Automatic verification of finite state concurrent systems using temporal logic specification: a practical approach. In *10th ACM Symposium on Principles of Programming Languages (POPL83)*. Complete version published in ACM TOPLAS, 8(2):244–263, April 1986.

10. E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. In *Symposium on Principles of Programming Languages (POPL 92)*. ACM, January 1992.

11. K. M. Chandy and J. Misra. *Parallel Program Design*. Addison-Wesley, Massachusetts, 1988.

12. D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving $\forall$CTL*, $\exists$CTL* and CTL*. In *IFIP Conference PROCOMET'94*. to appear, 1994.

13. P. Ernberg, L. Fredlund, and B. Jonsson. Specification and validation of a simple overtaking protocol using LOTOS. Technical Report T90006, SICS, Kista, Sweden, 1990.

14. E. A. Emerson and J. Y. Halpern. 'Sometimes' and 'not never' revisited: On branching versus linear time. In *10th ACM Symposium on Principles of Programming Languages (POPL 83)*. also published in Journal of ACM , 33:151-178.

15. O. Grumberg and E. Long. Compositionnal model checking and modular verification. In J.C.M. Baeten and J.F. Groote, editors, *Concur'91*, pages 250–265. LNCS 527, Springer-Verlag, 1991.

16. S. Graf and C. Loiseaux. Program verification using compositional abstraction. In *TAPSOFT 93, joint conference CAAP/FASE*. LNCS 668, Springer Verlag, April 1993.

17. S. Graf and C. Loiseaux. A tool for symbolic program verification and abstraction. In *Conference on Computer Aided Verification CAV 93, Heraklion Crete*. LNCS 697, Springer Verlag, 1993.

18. S. Graf. Verification of a distributed cache memory by using abstractions. In *Conference on Computer Aided Verification CAV'94, Stanford*. LNCS 818, Springer Verlag, June 1994.

19. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1984.

20. ISO. IS ISO/OSI 8807 - LOTOS: a formal description technique based on the temporal ordering of observational behaviour. International standard, ISO, 1989.

21. H. Jifeng. Various simulations and refinements. In *REX Workshop on Stepwise Refinement of Distributed Systems, Mook*. LNCS 430, Springer Verlag, 1989.

22. B. Jonsson. On decomposing and refining specifications of distributed systems. In *REX Workshop on Stepwise Refinement of Distributed Systems, Mook*. LNCS 430, Springer Verlag, 1989.

23. J. Katzelson and B. Kurshan. S/R: A Language for Specifying Protocols and other Coordinating Processes. In *5th Ann. Int'l Phoenix Conf. Comput. Commun.*, pages 286–292. IEEE, 1986.

24. D. Kozen. Results on the propositional μ-calculus. In *Theoretical Computer Science*. North-Holland, 1983.

25. R.P. Kurshan. Analysis of discrete event coordination. In *REX Workshop on Stepwise Refinement of Distributed Systems, Mook*. LNCS 430, Springer Verlag, 1989.

26. L. Lamport. The Temporal Logic of Actions. Technical Report 79, DEC, Systems Research Center, 1991.

27. C. Loiseaux. *Vérification symbolique de programmes réactifs à l'aide d'abstractions*. PhD thesis, February 1994.

28. N.A. Lynch and M.R. Tuttle. An introduction to Input/Output automata. Report MIT/LCS/TM 373, MIT, Cambridge, Massachussetts, November 1988.

29. R. Milner. An algebraic definition of simulation between programs. In *Proc. Second Int. Joint Conf. on Artificial Intelligence*, pages 481–489. BCS, 1971.

30. R. Milner. A calculus of communication systems. In *LNCS 92*. Springer Verlag, 1980.

31. R. Milner. A calculus for Synchrony and Asynchrony. *Journal of Theoretical Computer Science*, 25, 1983.

32. Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proceeding of 9th ACM Symposium on Principles of Distributed Computing*, 1990.

33. O. Ore. Galois connexions. *Trans. Amer. Math. Soc*, 55:493–513, February 1944.

34. A. Pnueli. The Temporal Logic of Programs. In *18th Symposium on Foundations of Computer Science (FOCS 77)*. IEEE, 1977. Revised version published in Theoretical Computer Science, 13:45–60, 1981.

35. A. Pnueli. Application of temporal logic to specification and verification of reactive systems: a survey of current trends. In *Current trends in Concurrency, Nordwijkerhout*. LNCS 224, Springer Verlag, 1986.

36. J.P. Queille. Le système CESAR: Description, spécification et analyse des applications réparties. Thesis, Université Scientifique et Médicale de Grenoble, June 1982.

37. Luis E. Sanchis. Data types as lattices: retractions, closures and projections. In *RAIRO Theorical computer science, vol 11, nr 4*, pages 339–344, 1977.

38. J. Sifakis. Property preserving homomorphisms and a notion of simulation of transition systems. RR 332, IMAG, Grenoble, November 1982.

39. J. Sifakis. Property preserving homomorphisms of transition systems. In E. Clarke and D. Kozen, editors, *4th Workshop on Logics of Programs, Pittsburgh*. LNCS 164, Springer Verlag, June 1983.

40. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56, 1983.