A Symbolic Semantics for Abstract Model Checking

Francesca Levi

Dipartimento di Informatica, Università di Pisa,
Corso Italia 40, 56100, Pisa, Italy,
levifran@di.unipi.it

Abstract

In this paper we present a symbolic semantics of value-passing concurrent processes where classical branching is replaced by separate relations of non-deterministic branch and alternative choice. The obtained symbolic graph is finite for regular processes and can suitably be interpreted over abstract values to effectively compute a safe abstract model for full μ-calculus model checking. The representation of non-determinism and alternative choice in symbolic transitions allows to achieve more precise approximations of the two dual next modalities.

Key words: Model Checking, μ-Calculus, Abstract Interpretation.

1 Introduction

Model checking is an automatic technique for verifying temporal properties of finite-state reactive systems, such as sequential circuits and communication protocols. In the last few years one of the main goals of the research on formal verification has been to extend the size of systems that can be effectively model checked. A basic contribution in this setting has been undoubtedly that of "symbolic" model checking [13], which allows to verify extremely large finite-state reactive systems by means of a BDD representation of the state-space. Another improvement with respect to classical model checking algorithms can be obtained with local (on-the-fly) algorithms [22,5] where a restricted part of the whole state-space is explored only. An orthogonal approach consists of reducing the size of the model to be verified by *abstraction*, namely by

[1] This work has been partially supported by the HCM project ABILE (ERBCHRXCT940624).

replacing the concrete model of the system with a smaller abstract model. Model checking on the abstract model can be used to deduce properties of the original system, whenever the abstract model satisfies only formulas that are satisfied by the concrete one (i.e. it is *safe*). The conjunction of abstraction and model checking is nowadays a very active area of research since it provides even for infinite systems a promising alternative to classical deductive systems, where complex proofs have to be constructed by hand.

Most of the recent proposals [3,11,14,4,20,10,17,15] are based on the theory of *abstract interpretation* [7,8] which was originally conceived in the framework of static analysis for designing approximate semantics of programs. The basic idea is that of obtaining an approximate semantics from the standard one by substituting the concrete domain of computation and its basic operations with an abstract simpler domain and corresponding safe abstract operations. In the model checking setting a formal abstract interpretation framework allows to formulate safeness of abstract models in a standard way as safeness of the corresponding approximate formulas semantics. More precisely an abstraction is safe whenever the semantics of formulas computed on the abstract model is a lower approximation of the concrete semantics of formulas. For branching-time logics the preservation of both existential and universal properties (or equivalently the preservation of the full logic with negation) gives some basic problems: negation is actually not monotonic. The combination of upper and lower approximations suggested by Kelb [14] provides an elegant way to deal with negation. Safeness for branching-time logics with explicit existential and universal modalities (and without negation) can equivalently be defined for abstract models as in [10,4] by considering two dual abstract transition relations: free abstract transitions to be used for approximating the universal next modality and constrained transitions to be used for approximating the existential next modality. The abstract model is safe whenever for every concrete transition there exists a free transition between the corresponding abstract states and for every constrained transition all the corresponding concrete transitions exist. Free and constrained abstract transitions induce exactly dual upper and lower approximations of the next modality.

Given a well understood notion of safe abstract model both for linear-time and full branching-time temporal logics the application in practice of abstract model checking requires to address the following main issue: how can a safe (and sufficiently precise) abstract model be effectively derived from the program without looking at the large (eventually infinite) concrete model? Ad-hoc techniques are mainly used in practice to build abstract models. By contrast classical abstract interpretation methodologies suggest to define systematic methods to directly build from programs approximations which are safe by construction. This approach is more general and allows furthermore to formally compare the precision of the results independently on the specific program.

In this paper we propose a systematic method for applying abstract interpretation to the μ-calculus model checking of value-passing concurrent processes. The basic contribution is the definition of a symbolic semantics of processes in the style of [12], which provides a finite representation of the classical infinite labelled transition system for regular processes. The main difference is that separate relations of non-determinism and alternative choice among transitions replace classical branching. Moreover, the infinite paths of [12] due to parameterized recursion are eliminated with a simple technique of generalization. The finite symbolic graph can be interpreted on concrete values to model check processes, namely to run-time compute the semantics of the logic over the concrete model. However, when processes are capable of exchanging values taken from an infinite set model checking on the symbolic graph is not effectively computable and an approximation is necessary to produce at least a partial answer. By the way an approximation is useful also for finite-state systems to tackle the classical state-explosion.

We propose a method to derive a safe abstract model by interpreting the symbolic graph on abstract values instead than on concrete values. The method is proved to be safe for full μ-calculus (with explicit negation) with respect to a formal abstract interpretation framework based on dual approximations (as in [14]); safeness of the underlying abstract model derives implicitly by safeness of the dual approximations of the formulas semantics. The basic approximation step concerns the next modality. We derive both a lower and an upper approximation by dual interpretations of symbolic transitions over abstract values corresponding to standard free and constrained transitions between abstract processes. For this purpose the explicit representation of non-deterministic and alternative choices is fruitfully exploited and allows us to achieve more precise dual abstract transitions than previous proposals [10].

The paper is organized as follows. Section 2 summarizes the basic concepts of abstract interpretation. Section 3 and 4 present value-passing concurrent processes, μ-calculus and concrete model checking. The symbolic graph is described in sections 5 and 6, while the corresponding model checking algorithm is shown in section 7. Sections 8 and 10 present abstract model checking. Related works are discussed in section 11.

2 Abstract Interpretation Theory

We briefly recall the basic ideas of abstract interpretation. More details can be found in [7–9]. The theory of abstract interpretation provides a systematic method to design approximate semantics of programs by replacing the concrete domain of computation with a simpler abstract domain. The relation between the concrete and the abstract domain is precisely stated into a formal framework, that allows to suitably express safeness and precision of approximations.

Definition 1 *Let* $(C, \leq)$ *and* $(A, \leq^{\#})$ *be two po-sets. A pair of functions* (α, γ)*, where* $\alpha : C \to A$ *(*abstraction*) and* $\gamma : A \to C$ *(*concretization*) is called a* Galois connection *iff* $\forall c \in C, \forall a \in A, \alpha(c) \leq^{\#} a \Leftrightarrow c \leq \gamma(a)$.

Orderings $\leq$ and $\leq^{\#}$ express precision in the corresponding domain. Thus, the condition $c \leq \gamma(\alpha(c))$ ensures the loss of information of abstraction to be safe. On the other hand, condition $\alpha(\gamma(a)) \leq^{\#} a$ ensures that no loss of information is due to concretization. These requirements can also be captured by saying that α is extensive ($c \leq \gamma(\alpha(c))$), γ is reductive ($\alpha(\gamma(a)) \leq^{\#} a$), α and γ are total and monotonic. If $\alpha(\gamma(a)) = a$, then (α, γ) is called a Galois insertion.

Suppose $\mathcal{S}(P)$ to be the semantics of a program P obtained as the least fix-point of a semantic function F over the concrete domain $(C, \leq)$. The idea is that of computing an approximate semantics $\mathcal{S}^{\#}(P)$ over the abstract domain $(A, \leq^{\#})$, that is a *safe* approximation of the concrete one. Safeness is suitably expressed by the condition $\alpha(\mathcal{S}(P)) \leq^{\#} \mathcal{S}^{\#}(P)$. The main result is that a safe approximate semantics $\mathcal{S}^{\#}(P)$ can be computed as the least fix-point of a safe approximate semantic function $F^{\#}$ over $(A, \leq^{\#})$.

Theorem 2 *Let* $(C, \leq)$ *and* $(A, \leq^{\#})$ *be the concrete and abstract domain and let* $F : C \to C$ *and* $F^{\#} : A \to A$ *be monotone functions. If for each* $c \in C$, $\alpha(F(c)) \leq^{\#} F^{\#}(\alpha(c))$*, then* $\alpha(\mathit{lfp}\ F) \leq^{\#} \mathit{lfp}\ F^{\#}$ *(*$F^{\#}$ *is a* safe *approximation of* F*).*

Given a Galois insertion there exist several safe approximations of the concrete function. The advantage of the abstract interpretation theory is that these approximations can suitably be compared by precision with respect to $\leq^{\#}$. Furthermore, there exists always a best (*optimal*) approximate semantic function $F^{\#}$, the one for which $\alpha(F(c)) = F^{\#}(\alpha(c))$ for each $c \in C$. In the Galois insertion setting safeness and optimality may equivalently be expressed as $F(\gamma(a)) \leq \gamma(F^{\#}(a))$ and $F(\gamma(a)) = \gamma(F^{\#}(a))$ for each $a \in A$.

3 The Processes

We consider a value-passing version of CCS depending on the domain of values *Val*, of variables *Var* and of channels *Chan*. Moreover, we assume a language of value expressions *Vexp* built from values and variables, ranged over by $e, \ldots$ and a language of boolean expressions *Bexp*, ranged over by $be, \ldots$.

Processes *Proc* are generated by the following grammar

$$P ::= nil \mid x \mid a.P \mid be \bigtriangledown P_1, P_2 \mid P_1 \times P_2 \mid P_1 + P_2 \mid P \setminus L \mid P(e_1, \ldots, e_n)$$

where $a \in \{b!e, b?x, \tau \mid b \in \mathit{Chan}, e \in \mathit{Vexp}, x \in \mathit{Var}\}$, $L \subseteq \mathit{Chan}$, $be \in \mathit{Bexp}$ is a boolean expression and $e_i \in \mathit{Vexp}$ are value expressions. Actions $b?x$ and $b!e$ are the receiving of a value v for x and the sending of the value of expression e, respectively. The operator $+$ represents choice, while $\times$ represents parallel composition. Process $be \bigtriangledown P_1, P_2$ behaves as P_1 if the expression be is evaluated to true, and as P_2 otherwise. The operator $\setminus L$ is the standard restriction for a set of channels L, while $P(e_1, \ldots, e_n)$ is a recursive call with parameters $e_1, \ldots, e_n$. We assume an associated definition $P(x_1, \ldots, x_n) \equiv T$ for each recursive process.

We say that an occurrence of a variable x is free in a process P if it does not lie within the scope of $c?x$. Therefore, we define $fv(P) = \{x \mid x$ is free in $P\}$, $bv(P) = \{x \mid x$ is not free in $P\}$ and $vars(P) = bv(P) \cup fv(P)$. Closed processes (with $fv(P) = \emptyset$) are ranged over by $p, \ldots$, while open processes are ranged over by $P, T \ldots$. In the following, a tuple of expressions $e_1, \ldots, e_n$ is denoted by $\bar{e}$ and $i \in \{1, \ldots, n\}$ is denoted by $i \in \{1, n\}$. We assume that each recursive definition $P(\bar{x}) \equiv T$ is such that $fv(T) \subseteq \{\bar{x}\}$ and that recursion is guarded in T.

A *substitution* is a partial function $\sigma : \mathit{Var} \to \mathit{VExp}$, where $tar(\sigma)$ and $dom(\sigma)$ denote its target and source, respectively. Process $P\sigma$ is obtained by substituting every free occurrence of x in P with $\sigma(x)$ for each $x \in dom(\sigma)$. In a similar manner substitutions can be extended to boolean and value expressions with respect to the given notion of free variables depending on the chosen languages *Vexp* and *Bexp*.

The behavior of closed processes is described by the standard transitions with labels $\mathit{Act} = \{\tau, b?v, b!v \mid b \in \mathit{Chan}, v \in \mathit{Val}\}$ of table 1, where we have omitted the symmetric rules for parallel composition and choice. The semantics depends on two functions to evaluate closed value and boolean expressions $\mathcal{S}_v : \mathit{Vexp} \to \mathit{Val}$ and $\mathcal{S}_b : \mathit{Bexp} \to \{tt, \mathit{ff}\}$. For $a \in \mathit{Act}$, $chan(\tau) = \emptyset$, $chan(b?v) = chan(b!v) = \{b\}$.

For a closed process $p \in \mathit{Proc}$, let $LTS(p) = (P^*, \overset{a}{\mapsto})$ be the labelled transition system, where $p \in P^*$ and for each $p' \in P^*$, if $p' \overset{a}{\mapsto} p''$ by the rules of table 1, $p'' \in P^*$. If the set of values is infinite, the labelled transition system is obviously infinite and infinitely branching.

Table 1

$$\tau.p \stackrel{\tau}{\mapsto} p \qquad\qquad c?x.p \stackrel{c?v}{\mapsto} p[v/x] \;\; v \in Val$$

$$c!e.p \stackrel{c!v}{\mapsto} p \;\; \mathcal{S}_v(e) = v \qquad \frac{p_1 \stackrel{a}{\mapsto} p_1'}{p_1 + p_2 \stackrel{a}{\mapsto} p_1'} +$$

$$\frac{p_1 \stackrel{a}{\mapsto} p_1'}{p_1 \times p_2 \stackrel{a}{\mapsto} p_1' \times p_2} \times_1 \qquad \frac{p_1 \stackrel{c!v}{\mapsto} p_1' \;\; p_2 \stackrel{c?v}{\mapsto} p_2'}{p_1 \times p_2 \stackrel{\tau}{\mapsto} p_1' \times p_2'} \times_2$$

$$\frac{p \stackrel{a}{\mapsto} p'}{p \setminus L \stackrel{a}{\mapsto} p' \setminus L} \setminus \; chan(a) \cap L = \emptyset \qquad \frac{T[\bar{e}/\bar{x}] \stackrel{a}{\mapsto} p'}{P(\bar{e}) \stackrel{a}{\mapsto} p'} rec \; P(\bar{x}) \equiv T$$

$$\frac{p_1 \stackrel{a}{\mapsto} p_1'}{be \bigtriangledown p_1, p_2 \stackrel{a}{\mapsto} p_1'} \bigtriangledown_1 \; \mathcal{S}_b(be) = tt \qquad \frac{p_2 \stackrel{a}{\mapsto} p_2'}{be \bigtriangledown p_1, p_2 \stackrel{a}{\mapsto} p_2'} \bigtriangledown_2 \; \mathcal{S}_b(be) = ff$$

4 The Logic

For expressing temporal properties of processes we consider a simple extension of propositional μ-calculus [16]. The main feature of μ-calculus is that classical temporal modalities are expressed by means of fix-points. Let $\mathcal{A}$ be a set of actions and VAR be a set of logical variables. Formulas are inductively defined as follows

$$A ::= X \mid A \wedge A \mid < K > A \mid \neg A \mid \mu X.A$$

where $X \in VAR$ is a logical variable and $K \subseteq \mathcal{A}$. The modality $< K >$ is a generalization of the classical existential next modality $< a >$ ranging over actions $a \in \mathcal{A}$ and it corresponds to $\vee_{a \in K} < a >$. The dual universal modality is $[K] \equiv \neg < K > \neg$. The operator $\mu X.A$ denotes the least fix-point and the dual operator of greatest fix-point is given by $\nu X.A \equiv \neg \mu X.A[\neg X/X]$. Operator $\mu X.$ acts as a binder for variable X and $FV(A) = \{X \mid X$ does not occur inside the scope of $\mu X.\}$. In order to ensure the semantics of formulas to be well-defined we require that each occurrence of a variable X is under an even number of negations.

Table 2
The semantics of μ-calculus

$$[\![X]\!]_\delta = \delta(X) \qquad [\![\neg A]\!]_\delta = \mathcal{S} \setminus [\![A]\!]_\delta$$
$$[\![A_0 \wedge A_1]\!]_\delta = [\![A_0]\!]_\delta \cap [\![A_1]\!]_\delta \qquad [\![\mu X.A]\!]_\delta = \mu V.([\![A]\!]_{\delta[V/X]})$$
$$[\![< K > A]\!]_\delta = [\![< K >]\!]([\![A]\!]_\delta)$$

Formulas are interpreted over a labelled transition system $\mathcal{M} = (\mathcal{S}, \overset{a}{\mapsto})$ with states $\mathcal{S}$ and transitions $s \overset{a}{\mapsto} s'$ with $a \in \mathcal{A}$. The set of processes able to perform a transition is obtained by means of the classical *next modality function* $[\![< K >]\!] : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$.

Definition 3 *For $K \subseteq \mathcal{A}$ and $S \in \mathcal{P}(\mathcal{S})$*

$$[\![< K >]\!](S) = \{s \in \mathcal{S} \mid \exists s \overset{a}{\mapsto} s',\ a \in K and\ s' \in S\}.$$

The rules to compute the set of states that satisfy an open formula are shown in table 2 with respect to a valuation $\delta : VAR \to \mathcal{P}(\mathcal{S})$ assigning subsets of states to logical variables. The valuation $\delta[V/X]$ agrees with δ except that $\delta[V/X](V) = \delta(X)$.

Given a labelled transition system $\mathcal{M} = (\mathcal{S}, \overset{a}{\mapsto})$, we say that a state satisfies a formula $s \models A$ ($s \in [\![A]\!]$) iff $s \in [\![A]\!]_\delta$ for each valuation δ. If A is a closed formula then $s \models A$ is independent on the valuation, namely $s \models A$ iff $s \in [\![A]\!]_\delta$ for some δ.

In order to express properties of value-passing processes we assume (without loss of expressiveness) that the set of actions K used in the next modality is either $K = \{\tau\}$, $K = \{b!v \mid v \in V, b \in Chan,\ V \subseteq Val\}$ or $K = \{b?v \mid v \in V, b \in Chan,\ V \subseteq Val\}$. In the following $K = \{b!v \mid v \in V$ such that $V \subseteq Val\}$ and $K = \{b?v \mid v \in V$ such that $V \subseteq Val\}$ are denoted by $b!V$ and $b?V$ respectively. With an abuse of notation singleton sets are written without brackets.

We write $< - > A \equiv \bigvee_{b \in Chan} \Big(< b?\,Val > A \vee < b!\,Val > A \Big) \vee < \tau > A$ (similarly for the dual $[-]A \equiv \bigwedge_{b \in Chan} \Big([b?\,Val]A \wedge [b!\,Val]A\Big) \wedge [\tau]A$).

Example 1 *Let $P(x) \equiv b!x.P(x+1)$ be a recursive process on the domain of natural numbers. Formula $\mu X. < b!n > true \vee [-]X$ is satisfied if the value $n \in Nat$ is eventually sent on channel b. By contrast, the property that always there are no values sent on channel a can be expressed by a formula $\nu X.[a!Nat]false \wedge [-]X$. A process $P(n')$ with $n' \leq n$ satisfies both properties.*

The idea of model checking is that of establishing whether $s \models A$ by exploring the labelled transition system. In the traditional global approach model checking is realized by exhaustively computing $[\![A]\!]$ with the rules of table 2.

5 The Symbolic Graph

Symbolic semantics [12] is a very popular model aiming at finitely representing the infinite labelled transition system of value-passing processes. The (eventually) finite model can successfully be exploited both to check bisimulation equivalence and to check temporal logic properties. The basic idea is to avoid input instantiation over values and to define transitions for open processes rather than for closed processes only.

A standard symbolic transition is $P \overset{(c,\theta)}{\mapsto} P'$ where P and P' are open processes, c is a constraint and θ is a symbolic action (i.e. an action possibly containing expressions with free variables). Such a transition represents the set of concrete transitions $P\rho \overset{a_\rho}{\mapsto} P'\rho$ for any assignment of values ρ to the free variables of P and P' such that the constraint c is satisfied. The corresponding concrete action a_ρ is obtained from θ by eventually evaluating the expressions with respect to the assigned values.

Example 2 *Let us consider a process $P \equiv b?x.P'$ where $P' \equiv x > 0 \bigtriangledown b!x + 1.P, a!x - 1.P$. Because of input instantiation the standard labelled transition system would be infinitely branching. The symbolic transitions describing the behavior of P are*

$$
\begin{aligned}
&P \overset{(true,b?x)}{\longrightarrow} P' \\
&P' \overset{(x>0,b!x+1)}{\longrightarrow} P \\
&P' \overset{(x\leq 0,a!x-1)}{\longrightarrow} P.
\end{aligned}
$$

The first transition represents all the concrete transitions of process P for all possible inputs on variable x. The resulting process can either send the value of $x+1$ on channel b or the value of $x-1$ on channel a depending on the value of x. As expressed by the constraints associated to symbolic actions $x+1$ is sent on channel b if $x > 0$ and $x - 1$ is sent on channel a otherwise.

We propose a version of symbolic transitions that differs from the classical one in some aspects. A main difference is that classical branching is replaced by separate relations of non-determinism ($\oplus$) and alternative choice ($\otimes$) among transitions. This representation provides useful information for the abstraction step. Intuitively the behaviour of all the closed processes obtained from an open process P is represented by a single transition

$$P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$$

where $\Theta_i = (c_i, \oplus_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ and $\Omega_i = \oplus_{j_i\in\{1,n_i\}}P_{i,j_i}$ for constraints c_i, symbolic actions θ_{i,j_i} and processes P_{i,j_i}. Alternative choices are related by $\otimes$ and non-deterministic choices by $\oplus$. The idea is that for each assignment of values to the free variables of P the behaviour of the corresponding closed process is modelled by exactly one alternative, the one for which constraint c_i is satisfied. All the concrete transitions of the resulting process are represented by the symbolic actions θ_{i,j_i} and by the corresponding processes P_{i,j_i} for each $j_i \in \{1, n_i\}$.

Before introducing the technical definitions let us show a simple example.

Example 3 *Let us consider the process of example 2* $P \equiv b?x.x > 0 \bigtriangledown b!x + 1.P, a!x - 1.P$. *The symbolic transitions describing the behavior of* P *are*

$$\begin{array}{l} P \stackrel{(true,b?x)}{\longrightarrow} P' \\ P' \stackrel{(x>0,b!x+1)\otimes(x\leq 0,a!x-1)}{\longrightarrow} P \otimes P. \end{array}$$

Process P' *has two alternatives depending on the value of variable* x *either* $x > 0$ *or* $x \leq 0$. *Therefore, the two possibilities are combined by* $\otimes$. *Suppose* P *to be* $b?x.(P' + d!x.P)$. *We have*

$$\begin{array}{l} P \stackrel{(true,b?x)}{\longrightarrow} P' + d!x.P \\ P' + d!x.P \stackrel{(x>0,b!x+1\oplus d!x)\otimes(x\leq 0,a!x-1\oplus d!x)}{\longrightarrow} (P \oplus P) \otimes (P \oplus P). \end{array}$$

Relation $\oplus$ *is used to model the non-deterministic choice between* $b!x + 1$ *and* $d!x$ *for the first alternative and between* $a!x - 1$ *and* $d!x$ *for the second one.*

Let us introduce some preliminary notions. A *simple substitution* is an injective function $\sigma : Var \to Var$. An *environment* is a total function $\rho : Var \to Val$. We denote by Sub and by Env the set of simple substitutions. and the set of environments. For an environment ρ we denote by $\rho[x \to v]$ the environment that agrees with ρ except that the value assigned to variable x is v. Let P be an open process and $x \in Var$. We say that x is *fresh* in P iff $x \notin vars(P)$. Moreover, we say that a process P is free for a simple substitution σ with $dom(\sigma) \subseteq fv(P)$, iff for each $x \in fv(tar(\sigma))$, $x \notin bv(P) \cup (fv(P) \setminus tar(\sigma))$. We denote by (P, ρ) the closed process $P\rho$.

We consider constraints with the following syntax.

Definition 4 *A* constraint *is*

$$c ::= be \mid e = e \mid true \mid e \in V \mid \neg c \mid c \wedge c \mid c \vee c$$

where $e \in Vexp$, $be \in Bexp$ and $V \subseteq Val$. The set of constraints is denoted by $\mathcal{C}$.

We extend in the obvious way to constraints the notions of free and bound variables $fv(c)$, $bv(c)$ and $vars(c)$. For $c \in \mathcal{C}$ we define the set of environments that satisfies a constraint $[\![c]\!] \in \mathcal{P}(Env)$ as $[\![be]\!] = \{\rho \in Env \mid \mathcal{S}_b(be\rho) = tt\}$, $[\![\neg c]\!] = Env \setminus [\![c]\!]$, $[\![c_1 \wedge c_2]\!] = [\![c_1]\!] \cap [\![c_2]\!]$, $[\![c_1 \vee c_2]\!] = \cup_{i\in\{1,2\}}[\![c_i]\!]$, $[\![e_1 = e_2]\!] = \{\rho \in Env \mid \mathcal{S}_v(e_1\rho) = \mathcal{S}_v(e_2\rho)\}$, $[\![e \in V]\!] = \cup_{v\in V}[\![e = v]\!]$. In the following, we write $\rho \models c$ for $\rho \in [\![c]\!]$ and we write $\models c$ for $\rho \in [\![c]\!]$ for each $\rho \in Env$.

Symbolic actions are formally defined as $SymAct = \{b?x, b!e, \tau \mid b \in Chan, x \in Var, e \in Vexp\}$ with $bv(b?x) = \{x\}$, $bv(b!e) = bv(\tau) = \emptyset$ and $fv(b!e) = vars(e)$, $fv(\tau) = fv(b?x) = \emptyset$.

Let us introduce the rules to build symbolic transitions.

5.1 Basic actions

Transitions of a basic process are obtained by the following rules.

$$a.P \xrightarrow{(true,a)} P \qquad a \in \{\tau, c!e\}$$

$$c?x.P \xrightarrow{(true,c?z)} P[z/x] \quad z \notin fv(c?x.P)$$

5.2 Choice

The rule of choice is based on the following ideas:

(1) an alternative choice for $P_1 + P_2$ is given by the conjunction of an alternative choice of P_1 and one of P_2;
(2) for every combination of alternative choices every non-deterministic choice of both P_1 and P_2 is possible.

The composition of two alternatives is defined as follows.

Definition 5 *Let* $\Theta_i = (c_i, \oplus_{j_i \in \{1,n_i\}} \theta_{i,j_i})$ *and* $\Omega_i = \oplus_{j_i \in \{1,n_i\}} P_{i,j_i}$ *for* $i \in \{1,2\}$. *We define*

$$\Theta_1 + \Theta_2 = (c_1 \wedge c_2, \oplus_{i \in \{1,2\}, j_i \in \{1,n_i\}} \tilde{\theta}_{i,j_i})$$
$$\Omega_1 + \Omega_2 = \oplus_{i \in \{1,2\}, j_i \in \{1,n_i\}} \tilde{P}_{i,j_i}$$

where for each $j_i \in \{1, n_i\}$,

(1) if $\theta_{i,j_i} \in \{\tau, c!e\}$, *then* $\tilde{\theta}_{i,j_i} = \theta_{i,j_i}$ *and* $\tilde{P}_{i,j_i} = P_{i,j_i}$;
(2) if $\theta_{i,j_i} = c?x$, *then* $\tilde{\theta}_{i,j_i} = c?z$ *and* $\tilde{P}_{i,j_i} = P_{i,j_i}[z/x]$, *for* $z \in$ *Var such that* $z \notin fv(c_h)$, *with* $h \in \{1,2\}$ *and* $h \neq i$, *and* P_{i,j_i} *is free for* $[z/x]$.

Let $\Theta_{i,j_i} = (c_{i,j_i}, \theta_{i,j_i,1} \oplus \ldots \oplus \theta_{i,j_i,k_{j_i}})$ and $\Omega_{i,j_i} = P_{i,j_i,1} \oplus \ldots \oplus P_{i,j_i,k_{j_i}}$ for $i \in \{1,2\}$ and $j_i \in \{1, n_i\}$:

$$\frac{\{P_i \overset{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{P_1 + P_2 \overset{\otimes_{j_i \in \{1,n_i\}} \Theta_{1,j_1} + \Theta_{2,j_2}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{1,j_1} + \Omega_{2,j_2}} +$$

For each pair of alternatives Θ_{1,j_1} and Θ_{2,j_2} with $j_1 \in \{1, n_1\}$ and $j_2 \in \{1, n_2\}$ $\Theta_{1,j_1} + \Theta_{2,j_2}$ is constrained by the conjunction of the constraints of Θ_{1,j_1} and Θ_{2,j_2} and it has as non-deterministic choices all possible actions of Θ_{1,j_1} and Θ_{2,j_2}. The corresponding $\Omega_{1,j_1} + \Omega_{2,j_2}$ is similarly obtained.

Example 4 *For a process* $P = b!x.P_1 + a!x.P_2$ *we have*

$$P \overset{(true, b!x \oplus a!x)}{\longrightarrow} P_1 \oplus P_2.$$

Process P *can actually non-deterministically perform* $b!x$ *and* $a!x$ *for each assignment to the free variables.*

5.3 Conditional

The rule of conditional is based on the following idea: an alternative for $be \bigtriangledown P_1, P_2$ is either an alternative of P_1, if *be* is satisfied, or an alternative of P_2, if *be* is not satisfied.

Definition 6 *Let* $\Theta = (c, \oplus_{j\in\{1,n\}}\theta_j)$ *and* $\Omega = \oplus_{j\in\{1,n\}} P_j$. *For* $c' \in \mathcal{C}$ *we define*

$$\Theta^{c'} = (c \wedge c', \oplus_{j\in\{1,n\}}\tilde{\theta}_j)$$
$$\Omega^{c'} = \oplus_{j\in\{1,n\}}\tilde{P}_j$$

where for each $j \in \{1, n\}$,

(1) if $\theta_j \in \{\tau, c!e\}$ *then* $\tilde{\theta}_j = \theta_j$ *and* $\tilde{P}_j = P_j$;
(2) if $\theta_j = c?x$ *then* $\tilde{\theta}_j = c?z$ *and* $\tilde{P}_j = P_j[z/x]$, *for* $z \in Var$ *such that* $z \notin fv(c')$ *and* P_j *is free for* $[z/x]$.

Let $\Theta_{i,j_i} = (c_{i,j_i}, \theta_{i,j_i,1} \oplus \ldots \oplus \theta_{i,j_i,k_{j_i}})$ and $\Omega_{i,j_i} = P_{i,j_i,1} \oplus \ldots \oplus P_{i,j_i,k_{j_i}}$ for $i \in \{1, 2\}$ and $j_i \in \{1, n_i\}$:

$$\frac{\{P_i \overset{\otimes_{j_i\in\{1,n_i\}}\Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i\in\{1,n_i\}}\Omega_{i,j_i}\}_{i\in\{1,2\}}}{be \bigtriangledown P_1, P_2 \overset{\otimes_{j_1\in\{1,n_1\}}\Theta^{be}_{1,j_1}\otimes_{j_2\in\{1,n_2\}}\Theta^{\neg be}_{2,j_2}}{\longrightarrow} \otimes_{j_1\in\{1,n_1\}}\Omega^{be}_{1,j_1} \otimes_{j_2\in\{1,n_2\}} \Omega^{\neg be}_{2,j_2}} \bigtriangledown$$

Constraint Θ^{be}_{1,j_1} is equivalent to Θ_{1,j_1} where the guard is additionally constrained by be, while $\Theta^{\neg be}_{2,j_2}$ is equivalent to Θ_{2,j_2} where the guard is additionally constrained by $\neg be$.

Example 5 *For* $P = x > 0 \bigtriangledown d!x.P_1 + a!x.P_2, b?x.P_3$ *we have* $P \overset{\Theta_1\otimes\Theta_2}{\longrightarrow} \Omega_1 \otimes \Omega_2$, *where* $\Theta_1 = (x > 0, d!x \oplus a!x)$, $\Theta_2 = (x \leq 0, b?z)$, $\Omega_1 = P_1 \oplus P_2$ *and* $\Omega_2 = P_3[z/x]$. *Note that the receive* $b?x$ *has to be renamed to a fresh variable in order to keep the variable distinct from the one of* $x \leq 0$.

5.4 Parallel Composition

The rule of parallel composition is quite complex, since a single rule has to realize at the same time synchronization and interleaving accordingly to $\oplus$ and $\otimes$. The definition is based on the following ideas:

(1) an alternative choice for $P_1 \times P_2$ is given by the conjunction of an alternative choice of P_1 and one of P_2;
(2) for every combination of alternative choices the non-deterministic choices of P_1 and P_2 have to be combined in parallel in all possible ways: both synchronization and interleaving have to be realized.

The non-deterministic choices corresponding to two alternatives are combined as follows.

Definition 7 *Let* $\Theta_i = (c_i, \oplus_{j_i \in \{1,n_i\}} \theta_{i,j_i})$ *and* $\Omega_i = \oplus_{j_i \in \{1,n_i\}} P_{i,j_i}$ *for* $i \in \{1,2\}$. *We define*

$$\Theta_1 \times \Theta_2 = (c_1 \wedge c_2, \oplus_{i \in \{1,2\}, j_i \in \{1,n_i\}} \theta_{1,j_1} \hat{\times} \theta_{2,j_2} \oplus_{i \in \{1,2\}, j_i \in \{1,n_i\}} \tilde{\theta}_{1,j_i})$$
$$\Omega_1 \times \Omega_2 = \oplus_{i \in \{1,2\}, j_i \in \{1,n_i\}} P_{1,j_1} \hat{\times} P_{2,j_2} \oplus_{j_i \in \{1,n_i\}, i \in \{1,2\}} \tilde{P}_{1,j_i}$$

where for each $j_i \in \{1, n_i\}$,

(1) $\theta_{1,j_1} \hat{\times} \theta_{2,j_2} = \tau$ *and* $P_{1,j_1} \hat{\times} P_{2,j_2} = P_{1,j_1} \times P_{2,j_2}[e/x]$, *if* $\theta_{1,j_1} = c!e$ *and* $\theta_{2,j_2} = c?x$;
(2) $\tilde{\theta}_{1,j_1} = \theta_{1,j_1}$ *and* $\tilde{P}_{1,j_1} = P_{1,j_1} \times P_2$, *if* $\theta_{1,j_1} \in \{\tau, c!e\}$;
(3) $\tilde{\theta}_{1,j_1} = c?z$ *and* $\tilde{P}_{1,j_1} = P_{1,j_1}[z/x] \times P_2$, *if* $\theta_{1,j_1} = c?x$ *and* $z \notin fv(P_2)$ *and* P_{1,j_1} *is free for* $[z/x]$.
(4) symmetric cases.

Let $\Theta_{i,j_i} = (c_{i,j_i}, \theta_{i,j_i,1} \oplus \ldots \oplus \theta_{i,j_i,k_{j_i}})$ and $\Omega_{i,j_i} = P_{i,j_i,1} \oplus \ldots \oplus P_{i,j_i,k_{j_i}}$ for $i \in \{1,2\}$ and $j_i \in \{1, n_i\}$:

$$\boxed{\dfrac{\{P_i \overset{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{P_1 \times P_2 \overset{\otimes_{j_i \in \{1,n_i\}} \Theta_{1,j_1} \times \ \Theta_{2,j_2}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{1,j_1} \times \ \Omega_{2,j_2}} \times}$$

For each pair of alternatives Θ_{1,j_1} and Θ_{2,j_2}, with $j_1 \in \{1, n_1\}$ and $j_2 \in \{1, n_2\}$, $\Theta_{1,j_1} \times \Theta_{2,j_2}$ has as a guard the conjunction of the constraints of Θ_{1,j_1} and Θ_{2,j_2} and it has as non-deterministic choices all possible combinations of the corresponding actions according to definition 7.

Example 6 *Consider an open process* $P_1 \times P_2$ *with* $P_1 = x > 0 \bigtriangledown d?x.T, a?x.T$ *and* $P_2 = d!(y+1).T + a!(y-1).T$ *for some process* T. *We have*

$$P_1 \overset{(x>0, d?z) \otimes (x \le 0, a?z)}{\longrightarrow} T[z/x] \otimes T[z/x]$$
$$P_2 \overset{(true, d!y+1 \oplus a!y-1)}{\longrightarrow} T \oplus T.$$

The transition resulting by the parallel composition is $P_1 \times P_2 \overset{\Theta_1 \otimes \Theta_2}{\longrightarrow} \Omega_1 \otimes \Omega_2$

where

$$\Theta_1 = (x > 0, \tau \oplus d?z \oplus d!y+1 \oplus a!y-1)$$
$$\Theta_2 = (x \leq 0, \tau \oplus a?z \oplus d!y+1 \oplus a!y-1)$$
$$\Omega_1 = T[y+1/x] \times T \oplus T[z/x] \times P_2 \oplus P_1 \times T \oplus P_1 \times T$$
$$\Omega_2 = T[y-1/x] \times T \oplus T[z/x] \times P_2 \oplus P_1 \times T \oplus P_1 \times T.$$

There are two alternative choices corresponding to constraints $x > 0$ and $x \leq 0$. The τ action of $x > 0$ arises from the synchronization of actions $d?z$ and $d!y+1$, while the one of $x \leq 0$ arises from $a?z$ and $a!y-1$. The other non-deterministic choices for each alternative are given by interleaving. The corresponding processes are obtained in the obvious way. For instance, the process corresponding to τ with guard $x > 0$ is $T[y+1/x] \times T$, since the value of $y+1$ has been received by P_1.

5.5 Recursion

Transitions of a recursive process are simply obtained by replacing the actual parameters to the formal parameters in the body.

$$\boxed{\frac{T[\bar{e}/\bar{x}] \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i}{P(\bar{e}) \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i}\ rec \quad P(\bar{x}) \equiv T}$$

5.6 Restriction

Definition 8 *Let $\Theta_i = (c_i, \oplus_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ and $\Omega_i = \oplus_{j_i\in\{1,n_i\}}P_{i,j_i}$ for $i \in \{1,n\}$. For $L \subseteq Chan$ we define $H_i^L = \{j_i \in \{1,n_i\} \mid chan(\theta_{i,j_i}) \notin L\}$ and $K^L = \{i \in \{1,n\} \mid$ there exists $j_i \in \{1,n_i\}$ such that $j_i \in H_i^L\}$. Moreover, $\Theta_i^L = (c_i, \oplus_{j_i\in H_i^L}\theta_{i,j_i})$ and $\Omega_i^L = \oplus_{j_i\in H_i^L}P_{i,j_i}$.*

$$\boxed{\frac{P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i}{P \setminus L \stackrel{\otimes_{i\in K^L}\Theta_i^L}{\longrightarrow} \otimes_{i\in K^L}\Omega_i^L}\ \setminus}$$

Soundness and completeness of the symbolic semantics are stated by the following theorem. This result shows both that classical transitions of table 1 are safely represented by symbolic transitions and that non-deterministic and alternative choices are properly composed by $\oplus$ and $\otimes$.

Lemma 9 *Let $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ be a symbolic transition, where $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ and $\Omega_i = \otimes_{j_i\in\{1,n_i\}}P_{i,j_i}$. For each $i \in \{1,n\}$ and $j_i \in \{1,n_i\}$, $fv(P_{i,j_i}) \subseteq fv(P) \cup bv(\theta_{i,j_i})$, $fv(c_i) \subseteq fv(P)$ and $bv(\theta_{i,j_i}) \cap fv(c_i) = \emptyset$.*

PROOF. By induction and by definitions 7, 6 and 5. $\square$

For $p_1, p_2 \in Proc$ be processes, we say that p_1 and p_2 are equivalent $p_1 \equiv p_2$ iff for each $a \in Act$, for each $p_1 \stackrel{a}{\mapsto} p_1'$ there exists $p_2 \stackrel{a}{\mapsto} p_2'$ and $p_1' \equiv p_2'$.

Theorem 10 *Let $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ be a symbolic transition, where $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ and $\Omega_i = \otimes_{j_i\in\{1,n_i\}}P_{i,j_i}$. For each environment $\rho \in Env$ there exists one and only one $i \in \{1,n\}$ such that $\rho \models c_i$ and*

(1) for each $j_i \in \{1,n_i\}$, if $\theta_{i,j_i} = c?x$ then $(P,\rho) \stackrel{c?v}{\mapsto} p$ such that $p \equiv (P_{i,j_i}, \rho[x \to v])$ for $v \in Val$, if $\theta_{i,j_i} = \tau$ then $(P,\rho) \stackrel{\theta_{i,j_i}}{\mapsto} p$ such that $p \equiv (P_{i,j_i}, \rho)$, and if $\theta_{i,j_i} = c!e$ then $(P,\rho) \stackrel{c!v}{\mapsto} p$ such that $p \equiv (P_{i,j_i}, \rho)$ and $\rho \models e = v$ for $v \in Val$;
(2) for each $(P,\rho) \stackrel{a}{\mapsto} p$ there exists $j_i \in \{1,n_i\}$ such that, if $a = c?v$ then $\theta_{i,j_i} = c?x$ and $(P_{i,j_i}, \rho[x \to v]) \equiv p$, if $a = \tau$ then $\theta_{i,j_i} = a$ and $(P_{i,j_i}, \rho) \equiv p$, and if $a = c!v$ then $\theta_{i,j_i} = c!e$ and $(P_{i,j_i}, \rho) \equiv p$, where $\rho \models e = v$.

Proof of theorem 10 is shown in the appendix.

6 A Finite Symbolic Graph

The rules of section 5 allow us to obtain a symbolic graph, that can be interpreted over environments to obtain the classical labelled transition system as proved by theorem 10. Unfortunately the naive application of the rule for recursion leads to an infinite symbolic graph even for regular processes, where there are no recursive calls inside parallel composition. The semantics of [12] suffers from the same problem.

Example 7 *Let us consider the process $P(x) \equiv b!x.P(x+1)$. Since the recursive process is unfolded infinitely often with a different argument an infinite* *graph arises.*

$$
\begin{aligned}
&P(x) \stackrel{(true, b!x)}{\longrightarrow} P(x+1)\\
&P(x+1) \stackrel{(true, b!x+1)}{\longrightarrow} P(x+1+1)\\
&\ldots
\end{aligned}
$$

We propose a method to avoid those infinite branches so that of obtaining a finite graph at least for regular processes. The idea to obtain a finite but equivalent symbolic graph is based on generalizing processes with the introduction of fresh variables in current recursive calls before constructing the transitions. Suppose to have built a transition $P \stackrel{(c,\theta)}{\mapsto} P'$; instead of constructing the transitions of P' we construct the transitions of a process obtained from P' by generalizing current recursive calls. This way rule *rec* cannot anymore be applied to infinite instances of the same process with different parameters in recursive calls. On the other hand, the obtained finite graph is able to correctly represent the behaviour of processes since the generalization step does not introduce loss of information. We call such a processes with current recursive calls with variables only *general processes.*

Example 8 *Let us consider the process $P(x)$ of example 7. We obtain the following finite graph*

$$P(x) \stackrel{(true, b!x)}{\longrightarrow} P(x+1)$$

The transitions of process $P(x+1)$ leading to an infinite graph are not constructed, process $P(x+1)$ can be actually generalized to $P(x)$. The obtained graph is equivalent to the infinite one since the behavior of process $P(x+1)$ can be obtained from the one of the general process $P(x)$ by applying the proper substitution $[x+1/x]$ of parameters.

In order to formally define this technique the main definitions of general processes and of generalization have to be introduced.

Definition 11 *We say that a process $P \in Proc$ is a* general process *if and only if $P \in \{nil, a.P', P(\bar{x}), P_1 + P_2, be \bigtriangledown P_1, P_2, P_1 \times P_2, P'' \setminus L$, such that P_1, P_2, P'' are general processes $\}$. The set of general processes is denoted by $\mathcal{GP}$.*

Since there are no current recursive calls a process $a.P'$ is a general process for any P'. On the other hand, a recursive process is general only if the parameters are variables. For the other process constructors the property of recursive calls has to be checked inductively on the single components.

For a general process $GP \in \mathcal{GP}$, let the *recursion variables* be $rv(a.P) = rv(nil) = \emptyset$, $rv(P(\bar{x})) = \{\bar{x}\}$, $rv(GP_1 + GP_2) = rv(GP_1 \times GP_2) = rv(be \bigtriangledown GP_1, GP_2) = rv(GP_1) \cup rv(GP_2)$ and $rv(GP \setminus L) = rv(GP)$.

The goal is that of representing a process P by a general process GP that correctly models its behavior. To this aim we introduce *most general processes.*

Definition 12 *Let $P \in Proc$ be a process. We define the* most general processes *of P as $\Pi(P) \subseteq \mathcal{GP}$:*

- $\Pi(P) = \{P\}$ *for* $P \in \{nil, a.P'\}$*;*
- $GP_1 + GP_2 \in \Pi(P_1 + P_2)$*,* $GP_1 \times GP_2 \in \Pi(P_1 \times P_2)$ *and* $be \bigtriangledown GP_1, GP_2 \in \Pi(be \bigtriangledown P_1, P_2)$*, for* $GP_i \in \Pi(P_i)$ *such that,* $rv(GP_1) \cap fv(GP_2) = \emptyset$*,* $rv(GP_1) \cap bv(GP_2) = \emptyset$ *and vice-versa;*
- $P(\bar{x}) \in \Pi(P(\bar{e}))$*, if* $P(\bar{z}) \equiv T$ *and* T *is free for* $[\bar{x}/\bar{z}]$*.*

Most general processes are simply obtained by introducing fresh and distinct variables in current recursive calls.

Example 9 *Let us consider the process of examples 7 and 8. Process $P(x)$ is a general process, while process $P(x+1)$ is not and $P(x)$ properly generalizes $P(x+1)$ i.e. $P(x) \in \Pi(P(x+1))$.*

The following properties are satisfied.

Proposition 13 *Let $P \in Proc$ be a process and $GP \in \Pi(P)$,*

(1) there exists one and only one substitution σ with $dom(\sigma) = rv(GP)$ such that $GP\sigma = P$.
(2) for each $GP' \in \Pi(P)$ there exists one and only one substitution σ' with $dom(\sigma') = rv(GP)$ and $tar(\sigma') = rv(GP')$ such that $GP\sigma' = GP'$.

Proposition 13 simply states that for $GP \in \Pi(P)$ there exists a substitution σ assigning to the formal parameters of recursive calls in GP the actual parameters of P so that $GP\sigma = P$. In the following, $GP \in \Pi(P)$ and $GP\sigma = P$ with $dom(\sigma) = rv(GP)$ is denoted by $GP \Longrightarrow_\sigma P$.

Most general processes are able to model the behavior of the corresponding processes as proved by proposition 14. Intuitively we have that for each environment ρ process (P, ρ) is equivalent to (GP, ρ') for a proper environment ρ'.

For $\rho \in Env$ and $\sigma \in Sub$, let $\rho \bigtriangleup \sigma = \rho' \in Env$ be the environment, such that $\rho'(x) = \rho(x)$ for $x \notin dom(\sigma)$, while $\rho'(x) = \mathcal{S}_v(\sigma(x)\rho)$ for $x \in dom(\sigma)$.

Proposition 14 *Let $P \in Proc$ be a process and $\rho \in Env$. For each $GP \Longrightarrow_\sigma P$, $(GP, \rho \bigtriangleup \sigma) \equiv (P, \rho)$.*

The environment $\rho \triangle \sigma$ is obtained by replacing in ρ the value of the variables in the domain of σ (the parameters of current recursive calls of GP) with the evaluation of the corresponding expressions over ρ (the corresponding actual parameters in P). Since $(GP, \rho \triangle \sigma)$ is equivalent to (P, ρ) the concrete transitions of (P, ρ) can be obtained from the symbolic transitions of GP provided that values are assigned to the recursion parameters accordingly to $\rho \triangle \sigma$.

By applying iteratively the generalization step before constructing the transitions we obtain a symbolic graph where transitions are $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ for general processes P.

Definition 15 *Let $P' \in Proc$ be a process. We define the* symbolic graph *$\mathcal{SG}(P') = (GP^*, T^*, \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$ with $GP^* \subseteq \mathcal{GP}$, $T^* \subseteq Proc$ and transitions $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ for each $P \in GP^*$ such that*

(1) $P' \in T^*$;
(2) for each $P \in T^*$ there exists $GP \in GP^* \cap \Pi(P)$;
(3) for each transition $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$, where $\Omega_i = \oplus_{j_i\in\{1,n_i\}}P_{i,j_i}$, $P_{i,j_i} \in T^*$ for $i \in \{1,n\}$ and $j_i \in \{1,n_i\}$.

Since the source state of transitions are general processes the symbolic graph is finite for regular processes.

Theorem 16 *Let $P \in Proc$ be a regular process. $\mathcal{SG}(P)$ is a finite graph up to renaming.*

Moreover, by proposition 14 and theorem 10 the symbolic graph is complete, namely it represents all processes and transitions of the standard labelled transition system.

Theorem 17 *For a closed process $p \in Proc$, let $LTS(p) = (P^*, \stackrel{a}{\mapsto})$ and let $\mathcal{SG}(p) = (GP^*, T^*, \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$. For each $p' \in P^*$, there exists $P \in GP^*$ and $\rho \in Env$ such that $p' \equiv (P, \rho)$.*

PROOF. By proposition 14 and theorem 10. □

Example 10 *Consider for instance the process $(P(1) \times Q(2)) \setminus \{b\}$, where $P(x) \equiv b!x.Q(x+1)$ and $Q(y) = b?z.P(z+y)$. All processes $P(e_1) \times Q(e_2)$ can be generalized to $P(w_1) \times Q(w_2)$ and all processes $Q(e_1) \times P(e_2)$ can be generalized to $Q(w_1) \times P(w_2)$ for fresh and distinct variables w_1 and w_2.*

Therefore, we have a finite graph

$$(P(w_1) \times Q(w_2)) \setminus \{b\} \stackrel{(true,\tau)}{\longrightarrow} (Q(w_1+1) \times P(w_1+w_2)) \setminus \{b\}$$
$$(Q(w_1) \times P(w_2)) \setminus \{b\} \stackrel{(true,\tau)}{\longrightarrow} (P(w_1+w_2) \times Q(w_2+1)) \setminus \{b\}.$$

This graph correctly describes the behavior of $(P(1) \times Q(2)) \setminus \{b\}$. For instance the concrete computation

$$(P(1) \times Q(2)) \setminus \{b\} \stackrel{\tau}{\mapsto} (Q(1+1) \times P(1+2)) \setminus \{b\} \stackrel{\tau}{\mapsto} \dots$$

is simulated by

$$(P(w_1) \times Q(w_1)) \setminus \{b\}, \rho_1) \stackrel{\tau}{\longrightarrow} \begin{array}{l} (Q(w_1+1) \times P(w_1+w_2)) \setminus \{b\}, \rho_1) \equiv \\ ((Q(w_1) \times P(w_2)) \setminus \{b\}, \rho_2) \stackrel{\tau}{\longrightarrow} \dots \end{array}$$

where $\rho_1(w_1) = 1$ and $\rho_1(w_2) = 2$, while $\rho_2 = \rho_1 \triangle [w_1 + 1/w_1, w_1 + w_2/w_2]$, i.e. $\rho_2(w_1) = 2$ and $\rho_2(w_2) = 3$. Environment ρ_2 assigns properly to parameters w_1 and w_2 the result of the evaluation of expressions $w_1 + 1$ and $w_1 + w_2$ with respect to ρ_1.

7 Model Checking on the Symbolic Graph

The symbolic graph can be interpreted over concrete values to construct the labelled transition system. Therefore, we define the model checking of formulas directly on the symbolic graph. Model checking on the symbolic graph cannot be computed for infinite set of values, but provides the basis to suitably derive abstract model checking from value abstraction.

Let $\mathcal{SG} = (GP^*, T^*, \stackrel{\otimes_{i \in \{1,n\}} \Theta_i}{\longrightarrow})$ be a symbolic graph and let $(\mathcal{P}(D), \subseteq)$ be the domain of closed processes such that $D = \{(P, \rho) \mid P \in GP^* \text{ and } \rho \in Env\}$. The semantics of formulas $[\![A]\!]^S$ over $(\mathcal{P}(D), \subseteq)$ is defined in table 3 with respect to symbolic valuations $\delta : VAR \to \mathcal{P}(D)$.

The main modification with respect to the standard semantics (table 2) concerns the next modality function $[\![< K >]\!]^S$. Let (P, ρ) be a process, a be an action and $S \subseteq D$. The problem is that of establishing if there exists a transition $(P, \rho) \stackrel{a}{\mapsto} (P', \rho)$ such that (P', ρ) belongs to S.

Concrete transitions are implicitly represented by symbolic transitions and can be derived by the conditions given by theorems 10 and 17 and proposition 14.

Table 3

$$[\![X]\!]^S_\delta = \delta(X) \qquad [\![\neg A]\!]^S_\delta = D \setminus [\![A]\!]^S_\delta$$

$$[\![A_0 \wedge A_1]\!]^S_\delta = [\![A_0]\!]^S_\delta \cap [\![A_1]\!]^S_\delta \qquad [\![\mu X.A]\!]^S_\delta = \mu V.([\![A]\!]^S_{\delta[V/X]})$$

$$[\![< K > A]\!]_\delta = [\![< K >]\!]^S([\![A]\!]^S_\delta)$$

Suppose for simplicity to have a symbolic transition such as $P \overset{(c,\theta)}{\mapsto} P'$. For an environment ρ there exists a corresponding transition $(P, \rho) \overset{a}{\mapsto} (P', \rho)$ if both $\rho \models c$ and action a agrees with the evaluation θ over ρ. These conditions can trivially be extended to deal with non-deteriministic and alternative choices. On the contrary, some care is necessary for checking if (P', ρ) belongs to S (where S is indeed the set of processes satisfying a formula): the domain contains actually only processes (GP, ρ) such that GP is a general process, while P' may be any process. Therefore, in order to have completeness it is necessary to check whenever (P', ρ) is contained in S up to generalization.

Both the conditions for the existence of transitions and for the check on the target process up to generalization are formalized as constraints.

Definition 18 *Let $K \in \{c!V, c?V, \tau\}$ and $\theta \in SymAct$. We define the constraint $\theta \preceq K$ as $\theta \preceq K \equiv true$, for $\theta = \tau$ and $\tau \in K$, $\theta \preceq K \equiv true$, for $\theta = c?x$ and $K = c?V$, $\theta \preceq K \equiv e \in V$, for $\theta = c!e$ and $K = c!V$, and $\theta \preceq K \equiv false$, otherwise.*

Constraint $\theta \preceq K$ is satisfied by an environment ρ iff the evaluation of θ in ρ is contained in the set of concrete actions K.

Definition 19 *Let $P \in \mathcal{GP}^*$ and $S \in \mathcal{P}(D)$. We define the constraint $P \preceq S$ as $\bigvee_{\{(GP,\rho') \in S | GP \Longrightarrow_\sigma P\}} \Big[\bigwedge_{x \in dom(\sigma)} \sigma(x) \in \rho'(x) \bigwedge_{x \notin dom(\sigma) \cap x \in fv(P)} x \in \rho'(x) \Big]$.*

Constraint $P \preceq S$ is satisfied by an environment ρ iff there exists $(GP, \rho') \in S$ such that $GP \Longrightarrow_\sigma P$ and $\rho \bigtriangleup \sigma = \rho'$.

Definition 20 *Let $P \overset{\otimes_{i \in \{1,n\}} \Theta_i}{\longrightarrow} \otimes_{i \in \{1,n\}} \Omega_i$ be a symbolic transition such that $\Theta_i = (c_i, \oplus_{j_i \in \{1,n_i\}} \theta_{i,j_i})$ and $\Omega_i = \oplus_{j_i \in \{1,n_i\}} P_{i,j_i}$. For $K \in \{c!V,\ c?V, \tau\}$ and $S \in \mathcal{P}(D)$ we define the constraint*

$$\psi(P, K, S) = \bigvee_{i \in \{1,n\}} \Big[c_i \wedge (\bigvee_{j_i \in \{1,n_i\}} (\theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq S)) \Big].$$

Intuitively a constraint $\psi(P, K, S)$ is satisfied by an environment ρ iff there exists an alternative c_i that is satisfied and for which there exists a non-deterministic choice such that both $\theta_{i,j_i} \preceq K$ and $P_{i,j_i} \preceq S$ hold. Therefore, if $\rho \models \psi(P, K, S)$ there exists a transition from (P, ρ) with action in K such that the resulting process is contained in S up to generalization. Such a constraint allows us to simply define the next modality function as follows.

Definition 21 *For $S \in \mathcal{P}(D)$ and $K \in \{c!V,\ c?V, \tau\}$ we define*

$$\begin{aligned}
&[\![< K >]\!]^S(S) = \{(P, \rho) \mid \ P \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i,\\
&\textit{either } \rho \models \psi(P, K, S) \ \ \textit{for } K \in \{\tau, c!V\},\\
&\textit{or } \rho[x \to v] \models \psi(P, K, S) \textit{ for } v \in V,\ \ K = c?V\}.
\end{aligned}$$

Model checking on the symbolic graph is indeed equivalent to model checking on the labelled transition system.

Theorem 22 *For each closed formula A, both $p \in [\![A]\!]$ implies $(P, \rho) \in [\![A]\!]^S$ for each $P \in GP^*$ and $\rho \in Env$ such that $p \equiv (P, \rho)$, and $(P, \rho) \in [\![A]\!]^S$ implies $p \in [\![A]\!]$ for each $p \in P^*$ such that $p \equiv (P, \rho)$.*

The proof of theorem 22 is shown in the appendix.

8 Abstract Model Checking

In this section we show the method to interpret the symbolic graph over abstract values in order to build (implicitly) a safe abstract model, i.e. a safe approximation of the semantics $[\![A]\!]^S$. The technique is defined into a formal abstract interpretation framework depending on the chosen value abstraction.

We assume the value abstraction to be given in a standard way by a Galois insertion (α_v, γ_v) between the domains $(\mathcal{P}(\mathit{Val}), \subseteq)$ and $(\mathcal{P}(\mathit{Val}^\#), \subseteq)$. The domains of abstract environments and processes are obtained accordingly by replacing concrete values with abstract values. Let $\mathcal{SG} = (GP^*, T^*, \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$ be a symbolic graph. The set of *abstract environments* is given by $\mathit{Env}^\# = \{\rho^\# \mid \rho^\# : \mathit{Var} \to \mathit{Val}^\#\}$ and the set of *abstract processes* is given by $D^\# = \{(P, \rho^\#) \mid P \in GP^*$ and $\rho^\# \in \mathit{Env}^\#\}$. Note that if the values domain $\mathit{Val}^\#$ is finite and GP^* is finite (which is the case for regular processes) then both $\mathit{Env}^\#$ and $D^\#$ are finite.

The abstraction of values naturally induces a relation between abstract and concrete environments as well as between abstract and concrete processes. An abstract environment intuitively represents the set of concrete environments such that a concrete value represented by the abstract value is assigned to each variable. Analogously for processes. In an abstract interpretation setting this property is formalized by concretization functions $\gamma_e : \mathcal{P}(Env^{\#}) \to \mathcal{P}(Env)$ and $\gamma : \mathcal{P}(D^{\#}) \to \mathcal{P}(D)$.

Definition 23 *Let $E^{\#} \in \mathcal{P}(Env^{\#})$ and $S^{\#} \in \mathcal{P}(D^{\#})$ we define γ_e and γ as follows*

$$\gamma_e(\{\rho^{\#}\}) = \{\rho \mid \rho(x) \in \gamma_v(\rho^{\#}(x)) \ \forall x \in Var\}$$
$$\gamma_e(E^{\#}) = \cup_{\rho^{\#} \in E^{\#}} \gamma_e(\{\rho^{\#}\})$$
$$\gamma(\{(P, \rho^{\#})\}) = \{(P, \rho) \mid \rho \in \gamma_e(\rho^{\#})\}$$
$$\gamma(S^{\#}) = \cup_{(P,\rho^{\#}) \in S^{\#}} \gamma(\{(P, \rho^{\#})\}).$$

Our goal is that of computing a safe lower approximation $[\![A]\!]^{\#}$ of the semantics $[\![A]\!]^{S}$ such that if an abstract process is proved to satisfy a property, then all concrete corresponding processes indeed satisfy the property. With respect to the concretization function γ safeness means that the abstract semantics is a lower approximation $\gamma([\![A]\!]^{\#}) \subseteq [\![A]\!]^{S}$. Following the basic principles of abstract interpretation (theorem 2) a safe lower approximation $[\![A]\!]^{\#}$ could be found by taking safe lower approximations for all the basic model checking functions corresponding to logical connectives. However, the operator of negation is not monotonic. A solution suggested by [14] is to obtain a lower approximation of the full logic with negation by combining duals approximations $[\![A]\!]^{l}$ and $[\![A]\!]^{u}$, such that $[\![A]\!]^{S} \subseteq \gamma([\![A]\!]^{u})$ and $\gamma([\![A]\!]^{l}) \subseteq [\![A]\!]^{S}$. We can define $[\![\neg A]\!]^{l} = D^{\#} \setminus [\![A]\!]^{u}$ which is a safe lower approximation ($[\![\neg A]\!]^{S} \supseteq \gamma([\![\neg A]\!]^{l})$), and $[\![\neg A]\!]^{u} = D^{\#} \setminus [\![A]\!]^{l}$ which is a safe upper approximation ($[\![\neg A]\!]^{S} \subseteq \gamma([\![\neg A]\!]^{u})$). This way the problem is reduced to the definition of safe dual approximations for all the logical operators except negation. The dual approximations are formally defined with respect to the following framework.

Proposition 24 *Let $\gamma : (\mathcal{P}(D^{\#}), \subseteq) \to (\mathcal{P}(D), \subseteq)$ be the function of definition 23. There exist $\alpha^{l}, \alpha^{u} : \mathcal{P}(D) \to \mathcal{P}(D^{\#})$ such that (α^{u}, γ) is a* Galois insertion *between $(\mathcal{P}(D), \subseteq)$ and $(\mathcal{P}(D^{\#}), \subseteq)$ and (α^{l}, γ) is a* Galois insertion *between $(\mathcal{P}(D), \supseteq)$ and $(\mathcal{P}(D^{\#}), \supseteq)$.*

PROOF. By standard abstract interpretation results [9]. It is sufficient to consider for every $S \in \mathcal{P}(D)$, $\alpha^{u}(S) = \cap\{S^{\#} \in \mathcal{P}(D^{\#}) \mid S \subseteq \gamma(S^{\#})\}$ and $\alpha^{l}(S) = \cup\{S^{\#} \in \mathcal{P}(D^{\#}) \mid \gamma(S^{\#}) \subseteq S\}$.

It is worth mentioning that the definition of the dual abstract interpretation

Table 4
For $\# \in \{l, u\}$ and $\hat{\#} = l$ for $\# = u$ and vice-versa

$$[\![X]\!]^{\#}_{\delta\#} = \delta^{\#}(X) \qquad [\![\mu X.A]\!]^{\#}_{\delta\#} = \mu V.([\![A]\!]^{\#}_{\delta\#[V/X]})$$
$$[\![A_0 \wedge A_1]\!]^{\#}_{\delta\#} = [\![A_0]\!]^{\#}_{\delta\#} \cap [\![A_1]\!]^{\#}_{\delta\#} \qquad [\![\neg A]\!]^{\#}_{\delta\#} = D^{\#} \setminus ([\![A]\!]^{\hat{\#}}_{\delta\#})$$
$$[\![< K > A]\!]^{\#}_{\delta\#} = [\![< K >]\!]^{\#}([\![A]\!]^{\#}_{\delta\#})$$

frameworks is not essential to prove the safeness of abstract model checking. Safeness of the dual approximations could be actually defined with respect to the concretization function γ. However, functions α^u and α^l are necessary to reason about optimality and precision. The approximations $\alpha^u([\![A]\!])$ and $\alpha^l([\![A]\!])$ give actually the best upper and lower approximations with respect to the chosen value abstraction. To better explain the difference between upper and lower approximation let us show a simple example over the domain of values.

Example 11 *Suppose to have $Val^{\#} = \{even, odd\}$ where $\gamma_v(even) = \{n \in Nat \mid n \text{ is even}\}$ and $\gamma_v(odd) = \{n \in Nat \mid n \text{ is odd}\}$. Let $EVEN = \{n \in Nat \mid n \text{ is even}\}$ and $N \subseteq Nat$ such that there exist $n_1, n_2 \in N$ where n_1 is even and n_2 is odd. Some cases of safe lower and upper abstractions are for n even:*

$$\alpha^u(n) = even \qquad \alpha^l(n) = \emptyset$$
$$\alpha^u(EVEN) = even \qquad \alpha^l(EVEN) = even$$
$$\alpha^u(N) = \{even, odd\} \qquad \alpha^l(N) = \emptyset$$

In the following we show the method to derive the dual approximations over the symbolic graph with respect to abstract environments. Let $\delta^{\#} : VAR \rightarrow \mathcal{P}(D^{\#})$ be an abstract valuation, the dual approximations of the semantics of formulas are defined by the rules of table 4 with respect to the corresponding approximation of the next modality $[\![< K >]\!]^{\#}$ for $\# \in \{u, l\}$. In the following, for a closed formula A we use the notation $(P, \rho^{\#}) \models^{\#} A$ for $(P, \rho^{\#}) \in [\![A]\!]^{\#}_{\delta\#}$ for each valuation $\delta^{\#}$ and for $\# \in \{l, u\}$.

In order to define $[\![< K >]\!]^u$ and $[\![< K >]\!]^l$ we assume dual approximations for the evaluation of constraints $[\![c]\!]^u \in \mathcal{P}(Env^{\#})$ and $[\![c]\!]^l \in \mathcal{P}(Env^{\#})$ such that $[\![c]\!] \subseteq \gamma_e([\![c]\!]^u)$ and $\gamma_e([\![c]\!]^l) \subseteq [\![c]\!]$. In the following, $\rho^{\#} \models^{\#} c$ denotes $\rho^{\#} \in [\![c]\!]^{\#}$ for $\# \in \{u, l\}$.

The dual approximations of the next modality function require to compute constrained and free abstract transitions [10,4]) between abstract processes.

The upper approximation of the next modality function $[\![< K >]\!]^u$ is safe, if $(P, \rho) \in [\![< K >]\!]^S(\gamma(S^\#))$ for some $\rho \in \gamma_e(\rho^\#)$ implies $(P, \rho^\#) \in [\![< K >]\!]^u(S^\#)$. Therefore, in order to compute $[\![< K >]\!]^u$ at least the abstract transitions for which a corresponding concrete transition exists must be considered (free abstract transitions [10,4]). We define an evaluation of symbolic transitions over abstract values which respects this idea.

Definition 25 *For $S^\# \in \mathcal{P}(D^\#)$ and $K \in \{c!V,\ c?V, \tau\}$ we define*

$$\begin{aligned}
&[\![< K >]\!]^u(S^\#) = \{(P, \rho^\#) \mid\ P \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i,\\
&\textit{either } \rho^\# \models^u \psi(P, K, \gamma(S^\#)) \textit{ for } K \in \{\tau, c!V\},\\
&\textit{or } \rho^\#[x \to \alpha_v(v)] \models^u \psi(P, K, \gamma(S^\#)),\ \textit{for } v \in V,\ K = c?V\}.
\end{aligned}$$

The approximation is obtained by replacing in definition 21 the concrete evaluation of $\psi(P, K, S)$ with the upper approximate evaluation of $\psi(P, K, \gamma(S^\#))$. Safeness follows trivially by the safeness of constraints evaluation. Suppose that there exists a concrete transition for an environment $\rho \in \gamma_e(\rho^\#)$. By definition $\rho \models \psi(P, K, \gamma(S^\#))$ so that $\rho^\# \models^u \psi(P, K, \gamma(S^\#))$. Therefore, all concrete transitions for some $\rho \in \gamma_e(\rho^\#)$ give rise to an abstract free transition for $\rho^\#$. Note that the concrete set of processes $\gamma(S^\#)$ is considered in order to check the target process to be contained in $S^\#$ up to generalization. In the following we will discuss the possible implementation of this step.

Example 12 *Consider a process $P \equiv b!x.P'$ with variable x ranging over natural numbers. Suppose that natural numbers are abstracted in the standard way to values $\{even, odd\}$. We have two abstract environments $\rho_1^\#$ and $\rho_2^\#$ such that $\rho_1^\#(x) = even$ and $\rho_2^\#(x) = odd$. Since there exists $\rho \in \gamma_e(\rho_1^\#)$ such that $\rho(x) = 2$ we have $\rho_1^\# \models^u true \wedge x = 2 (\equiv \psi(P, b!2, D))$. Therefore we have $(P, \rho_1^\#) \models^u < b!2 > true$.*

Example 13 *Consider a process $P \equiv x > 0 \bigtriangledown b?x.P_1, a?x.P_2$, whose behavior is described by*

$$P \overset{(x>0,b?x)\otimes(x\leq 0,a?x)}{\longrightarrow} P_1 \otimes P_2.$$

Suppose the chosen abstract domain of abstract values to be $Val^\# = \{\top\}$ with $\gamma_v(\top) = Val$. Let $\rho^\#$ be the abstract environment with $\rho^\#(x) = \top$. We obtain both $(P, \rho^\#) \models^u < b?v > true$ and $(P, \rho^\#) \models^u < a?v > true$, since $\rho^\# \models^u x > 0$ $(\equiv \psi(P, b?n, D))$ and $\rho^\# \models^u x \leq 0$ $(\equiv \psi(P, a?n, D))$. There exist actually

$\rho_1, \rho_2 \in \gamma_e(\rho^\#)$ *such that*

$$\rho_1 \models (x > 0 \wedge (b?x = b?n)) \vee (x \leq 0 \wedge (a?x = c?n))$$
$$\rho_2 \models (x \leq 0 \wedge (a?x = a?n)) \vee (x > 0 \wedge (b?x = a?n)).$$

The abstract operator is safe, since the existence of both concrete transitions $(P, \rho_1) \overset{b?n}{\mapsto} (P_1, \rho_1[x \to n])$ *and* $(P, \rho_2) \overset{a?n}{\mapsto} (P_2, \rho_2[x \to n])$ *is captured.*

Lemma 26 *For each* $S^\# \in \mathcal{P}(D^\#)$ *and* $K \in \{c?V, c!V, \tau\}$,

$$\gamma([\![< K >]\!]^u(S^\#)) \supseteq [\![< K >]\!]^S(\gamma(S^\#)).$$

PROOF. We have to prove that, for each $P \in GP^*$ and $\rho \in Env$, if $(P, \rho) \in [\![< K >]\!]^S(\gamma(S^\#))$ then $(P, \rho^\#) \in [\![< K >]\!]^u(S^\#)$ for $\rho^\#$ such that $\rho \in \gamma_e(\rho^\#)$. By definition 21, $\rho \models \psi(P, K, \gamma(S^\#))$ so that by safeness of constraints evaluation $\rho^\# \models^u \psi(P, K, \gamma(S^\#))$. $\square$

8.2 Lower approximation

Safeness of the lower approximation of the next modality is more complex: $(P, \rho^\#) \in [\![< K >]\!]^l(S^\#)$ requires that $(P, \rho) \in [\![< K >]\!]^S(\gamma(S^\#))$ for each $\rho \in \gamma_e(\rho^\#)$. Therefore, in order to compute $[\![< K >]\!]^l$ only the abstract transitions for which all corresponding concrete transition exist must be considered (constrained abstract transitions [10,4]).

A naive solution to obtain a safe lower approximation $[\![< K >]\!]^l$ could be to consider the dual case of definition 25, where the upper approximation of constraint $\psi(P, K, \gamma(S^\#))$ is replaced by the lower approximation. Actually $\rho^\# \models^l \psi(P, K, \gamma(S^\#))$ guarantees that $\rho \models \psi(P, K, \gamma(S^\#))$ for each $\rho \in \gamma_e(\rho^\#)$. We propose a different solution that exploits more properly the relations $\otimes$ and $\oplus$.

Definition 27 *Let* $P \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ *be a symbolic transition with* $\Theta_i = (c_i, \oplus_{j_i\in\{1,n_i\}}\ \theta_{i,j_i})$ *and* $\Omega_i = \oplus_{j_i\in\{1,n_i\}} P_{i,j_i}$. *For* $K \in \{c!V,\ c?V, \tau\}$ *and* $S \in \mathcal{P}(D)$ *we define the constraint*

$$\psi^l(P, K, S) = \bigwedge_{i\in\{1,n\}} \Big[\neg c_i \vee (\bigvee_{j_i\in\{1,n_i\}} (\theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq S))\Big].$$

Intuitively, an environment ρ satisfies constraint $\psi^l(P, K, S)$, whenever for each alternative c_i either the guard is not satisfied or there exists a non-deterministic choice for which both action θ_{i,j_i} is in K and the resulting process is contained in S up to renaming.

Note 1 *Due to theorem 10* $\psi^l(P,K,S) \equiv \psi(P,K,S)$*, since for each environment there exists one and only one* c_i *that is satisfied. However, the lower approximation of the two constraints may lead to different results: the approximation of* $\psi^l(P,K,S)$ *may actually be more precise. We will show an example in section 9.*

Definition 28 *For* $S^\# \in \mathcal{P}(D^\#)$ *and* $K \in \{c!V,\ c?V, \tau\}$ *we define*

$$
\begin{aligned}
&[\![< K >]\!]^l(S^\#) = \{(P,\rho^\#) \mid \ P \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i,\\
&\textit{either } \rho^\# \models^l \psi^l(P,K,\gamma(S^\#)) \ \ \textit{for } K \in \{\tau, c!V\},\\
&\textit{or } \rho^\#[x \to \alpha_v(v)] \models^l \psi^l(P,K,\gamma(S^\#)) \textit{ for } v \in V,\ \ K = c?V\}.
\end{aligned}
$$

The definition has the following motivations. If $\rho^\# \models^l \psi^l(P,K,\gamma(S^\#))$, then for each environment $\rho \in \gamma_e(\rho^\#)$ and for each $i \in \{1,n\}$, $\rho \models c_i$ implies the existence of a non-deterministic choice $j_i \in \{1,n_i\}$ such that both $\rho \models \theta_{i,j_i} \preceq K$ and $\rho \models P_{i,j_i} \preceq \gamma(S^\#)$. Therefore, if $\rho \models c_i$ a required concrete transition with action in K and resulting process in $\gamma(S^\#)$ up to generalization exists. Since for any ρ it cannot be the case that $\rho \models \neg c_i$ for each $i \in \{1,n\}$ this condition guarantees that at least a concrete transition exists for each environment.

Example 14 *Consider the process* $P \equiv b!x.P'$ *of example 12, where natural numbers are abstracted to abstract values* $\{even, odd\}$ *in the obvious way. We have two abstract environments* $\rho_1^\#$ *and* $\rho_2^\#$ *such that* $\rho_1^\#(x) = even$ *and* $\rho_2^\#(x) = odd$*. We cannot prove* $(P,\rho_1^\#) \models^l < b!2 > true$*, since there exists* $\rho \in \gamma_e(\rho_1^\#)$ *with* $\rho(x) \neq 2$*. By the lower approximation of constraints actually* $\rho_1^\# \not\models^l false \vee x = 2$ $(\equiv (\psi^l(P,b!2,D))$.

Since $\rho_1^\# \models^l x \in \gamma_v(even)$ *we can prove a weaker property* $< c!\gamma_v(even) > true$*.*

Example 15 *Consider the process* $P \equiv x > 0 \bigtriangledown b?x.P_1, a?x.P_2$ *of example 13. Let the value abstraction be* $\gamma_v(\top) = Val$ *and let* $\rho^\#$ *be the abstract environment with* $\rho^\#(x) = \top$*. Since there exists* $\rho_1, \rho_2 \in \gamma_e(\rho^\#)$ *such that* $(P,\rho_1) \not\models < b?n > true$ *and* $(P,\rho_2) \not\models < a?n > true$ *the lower approximation is safe if and only if both* $(P,\rho^\#) \not\models^l < b?n > true$ *and* $(P,\rho^\#) \not\models^l < a?n > true$*.*

We have indeed $\rho^\# \not\models^l \psi^l(P,b?n,D)$ *for* $\psi^l(P,b?n,D) \equiv (x \leq 0 \vee (b?x = b?x)) \wedge (x > 0 \vee false)$ *since there exist concrete environments for both alternatives that are abstracted to* $\rho^\#$*. When* $x \leq 0$ *an action* $b?n$ *cannot be performed by the corresponding process. The dual case is similar.*

On the other hand, consider the abstract domain of values $\{Pos, Neg\}$ such that $\gamma_v(Pos) = \{n \mid n > 0\}$ and $\gamma_v(Neg) = \{n \mid n \leq 0\}$. There are two abstract environments $\rho_1^\#$ with $\rho_1^\#(x) = Pos$ and $\rho_2^\#$ with $\rho_2^\#(x) = Neg$. Since for each $\rho \in \gamma(\rho_1^\#)$, $\rho \models \psi^l(P, b?n, D)$ then $\rho_1^\# \models^l \psi^l(P, b?n, D)$ is safe. Therefore, $(P, \rho_1^\#) \models^l < b?n >$ true as well as $(P, \rho_2^\#) \models^l < a?n >$ true can be proved. For all environments abstracted to $\rho_1^\#$ (resp. $\rho_2^\#$) actually only the alternative $x > 0$ (resp. $x \leq 0$) is possible.

Lemma 29 *For each $S^\# \in \mathcal{P}(D^\#)$ and $K \in \{\tau, c?V, c!V\}$,*

$$\gamma([\![< K >]\!]^l(S^\#)) \subseteq [\![< K >]\!]^S(\gamma(S^\#)).$$

PROOF. We have to prove that, for each $P \in GP^*$ and $\rho^\# \in Env^\#$, $\rho^\# \in [\![< K >]\!]^l(S^\#)$ implies $\rho \in [\![< K >]\!]^S(\gamma(S^\#))$ for each $\rho \in \gamma_e(\rho^\#)$. By definition 28 $\rho^\# \models^l \psi^l(P, K, \gamma(S^\#))$. Therefore, by safeness of constraints evaluation for each $\rho \in \gamma_e(\rho^\#)$, $\rho \models \psi^l(P, K, \gamma(S^\#))$. By theorem 10 for each $\rho \in Env$ there exists one and only one $i \in \{1, n\}$, such that $\rho \models c_i$. Therefore, for each $\rho \in \gamma_e(\rho^\#)$ there exists $i \in \{1, n\}$ such that both $\rho \models c_i$ and $\rho \models \bigvee_{j_i \in \{1, n_i\}} (\theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq \gamma(S^\#))$. Thus $\rho \models \bigvee_{i \in \{1,n\}} \left[c_i \vee (\bigvee_{j_i \in \{1,n_i\}} (\theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq \gamma(S^\#)))\right] \equiv \psi(P, K, \gamma(S^\#))$. By definition 21 for each $\rho \in \gamma_e(\rho^\#)$, $(P, \rho) \in [\![< K >]\!]^S(\gamma(S^\#))$. $\square$

9 Safeness and Precision of Abstract Model Checking

Safeness of abstract model checking for full μ-calculus follows from lemma 26, 29 and theorem 2.

Theorem 30 *For each closed μ-calculus formula A, $[\![A]\!]^S \supseteq \gamma([\![A]\!]^l)$.*

The precision of the approximate semantics mainly depends on the precision of the dual approximations of $[\![< K >]\!]^S$ which rely on dual approximations of simple constraints. The approximate semantics is in general non-optimal even if optimal approximations of constraints evaluation would be available. The intersection $\cap$ and the union $\cup$ typically lose optimality in the upper and lower approximation respectively.

However, the method proposed allows us to achieve a more precise lower approximation with respect to other approaches. In particular, the relations $\otimes$ and $\oplus$ are fruitfully exploited to achieve a more precise lower approximation (constrained abstract transition relation). The standard approach to achieve a safe lower approximation $[\![< K >]\!]^l$ consists of considering the dual case of

$[\![< K >]\!]^u$. The definition would indeed be safe, but less precise as formally proved by the following result.

Proposition 31 *Let $K \in \{c?V, c!V, \tau\}$, let $S \in \mathcal{P}(D)$ and $P \in Proc$. We have $[\![\psi(P, K, S)]\!]^l \subseteq [\![\psi^l(P, K, S)]\!]^l$.*

The weakness of $\psi(P, a, \gamma(S^{\#}))$ with respect to $\psi^l(P, a, \gamma(S^{\#}))$ is well-explained by the following example.

Example 16 *Consider a process $P = d?x.(P_1(x) \times P_2) \setminus \{a, b\}$ with $P_1(x) = x > 0 \bigtriangledown b!x.P(x+1), a!x.P(x-1)$ and $P_2 = b?x.P_2 + a?x.P_2$. The behavior is described by the symbolic graph*

$$P \overset{(true, d?x)}{\longrightarrow} (P_1(x) \times P_2) \setminus \{a, b\}$$
$$(P_1(x) \times P_2) \setminus \{a, b\} \overset{\Theta_1 \otimes \Theta_2}{\longrightarrow} \Omega_1 \otimes \Omega_2$$

where $\Theta_1 = (x > 0, \tau)$, $\Theta_2 = (x \leq 0, \tau)$, $\Omega_1 = P_1(x+1) \times P_2$ and $\Omega_2 = P_1(x-1) \times P_2$.

Suppose to consider the trivial abstract domain $Val^{\#} = \{\top\}$ with $\gamma_v(\top) = Val$. Let $\rho^{\#}$ be the abstract environment such that such that $\rho^{\#}(x) = \top$. We are able to prove $\rho^{\#} \models^l < \tau > true$, since $\rho^{\#} \models^l \psi^l(P, \tau, D)$ where $\psi^l(P, \tau, D) \equiv (x \leq 0 \vee (\tau = \tau)) \wedge (x > 0 \vee (\tau = \tau)) \equiv true$. Constraint $\psi^l(P, \tau, D)$ is satisfied by $\rho^{\#}$ even if $\rho^{\#}$ represents processes for both alternatives: action τ can be actually performed by both alternatives.

By contrast, $\rho^{\#} \not\models^l (x > 0 \wedge (\tau = \tau)) \vee (x \leq 0 \wedge (\tau = \tau)) (\equiv \psi(P, \tau, D))$, since both $\rho^{\#} \not\models^l (x > 0 \wedge (\tau = \tau))$ and $\rho^{\#} \not\models^l (x \leq 0 \wedge (\tau = \tau))$. The problem is that in the abstract setting it is not true that either $\rho^{\#} \models^l x > 0$ or $\rho^{\#} \models^l x \leq 0$: $\rho^{\#}$ may represent actually both environments satisfying $x > 0$ and environments satisfying $x \leq 0$. Therefore, when environments corresponding to different alternatives are abstracted to the same abstract environment the previous approach may be less powerful for computing abstract constrained transitions.

Since $\rho^{\#} \models^l < \tau > true$, then $[\![\mu X. \neg < \tau > true \vee < - > X]\!]^u = \emptyset$. Therefore, we have

$$[\![< d?Val > (\mu X. \neg < \tau > true \vee < - > X)]\!]^u = \emptyset$$
$$(P, \rho^{\#}) \in [\![\neg(< d?Val > (\mu X. \neg < \tau > true \vee < - > X))]\!]^l.$$

Due to the improved lower approximation of the next modality, we are able to prove that for each value received on channel d always the process is able *to perform a silent action. This property could not be proved in the simple approach.*

10 Approximating Constraints

We have proposed a systematic method of abstract model checking, where the approximate next modalities rely on approximations of constraints (definitions 28 and 25). In this section we discuss some approaches for realizing in practice the dual constraints approximations assuming the abstract set of values to be finite. Specific solutions should be studied once the domains of value and boolean expressions have been chosen.

The constraints which have to be approximated for model checking are the following. Let $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ be a symbolic transition with $\Theta_i = (c_i, \oplus_{j_i\in\{1,n_i\}}\ \theta_{i,j_i})$ and $\Omega_i = \oplus_{j_i\in\{1,n_i\}}P_{i,j_i}$. For $K \in \{c!V,\ c?V, \tau\}$ and $S \in \mathcal{P}(D)$ we have to consider

(1) $\psi^l(P,K,S) = \bigwedge_{i\in\{1,n\}}\left[\neg c_i \vee (\bigvee_{j_i\in\{1,n_i\}}(\theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq S))\right]$;
(2) $\psi(P,K,S) = \bigvee_{i\in\{1,n\}}\left[c_i \wedge (\bigvee_{j_i\in\{1,n_i\}}(\theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq S))\right]$.

The first problem is that infinite disjunctions $e \in V$ (arising from the evaluation of a modality $< c!V >$ for instance) and $P \preceq \gamma(S^{\#})$ (arising from generalization) may appear in constraints. Therefore, methods to effectively deal with those infinite sets have to be found out. A possibility is that of finitely representing these constraints.

A constraint $c \in \mathcal{C}$ is said *finite* if it does not contain occurrence of $e \in V$ such that $V \subseteq Val$ is an infinite set.

Definition 32 *An infinite set of values $V \subseteq Val$ is* representable *if there exists a finite constraint $c \in C$ such that $fv(c) = \{x\}$ and $V = \{v \in Val \mid \models c[v/x]\}$. A concretization function $\gamma_v : \mathcal{P}(Val^{\#}) \to \mathcal{P}(Val)$ is* representable *if there exists a finite set of finite constraints C such that for each $v^{\#} \in Val^{\#}$ the set of abstract values $\gamma_v(v^{\#})$ is* representable *by a constraint $c \in C$.*

Constraints $e \in V$ such that V is an infinite but representable set of values can be replaced by an equivalent finite constraint. If the concretization function is representable also a constraint $P \preceq \gamma(S^{\#})$ can be replaced by an equivalent finite constraint (assuming $S^{\#}$ to be finite which is trivially true whenever $Val^{\#}$ is finite).

Let $\rho^{\#} \in Env^{\#}$ be an abstract environment and let C be the set of constraints that represent γ_v. We denote by $\rho^{\#}_x$ the constraint $c \in C$ with $fv(c) = \{x\}$

that represents $\gamma_v(\rho^{\#}(x))$, for each $x \in \mathit{Var}$.

Proposition 33 *Let $V \subseteq \mathit{Val}$ be an infinite set of values representable by a finite constraint c and let γ_v be a concretization function* representable *by the set of constraints C. For each $e \in \mathit{VExp}$, $P \in \mathit{Proc}$ and $S^{\#} \in \mathcal{P}(D^{\#})$ there exist finite constraints $c_1 \equiv e \in V$ and $c_2 \equiv P \preceq \gamma(S^{\#})$.*

PROOF. It is sufficient to consider $c_2 \equiv c[e/x]$ such that $fv(c) = \{x\}$ and $c_2 \equiv \bigvee_{\{(GP,\rho^{\#}) \in S^{\#} | GP \Longrightarrow_{\sigma} P\}} \left[\bigwedge_{x \in dom(\sigma)} \rho^{\#}_x[\sigma(x)/x] \bigwedge_{x \notin dom(\sigma) \cap x \in fv(P)} \rho^{\#}_x \right]$. $\square$

Therefore, abstract model checking deals with approximations of finite constraints only, whenever the concretization function is representable and the set of values in the next modalities of the formula to be proved are representable. More powerful (for instance first-order) domains of constraints could be alternatively used to represent infinite constraints arising during model checking. In such a case the dual abstractions of constraints evaluation have to be defined for the new domains.

The second problem is that of effectively approximating finite constraints. Suppose that the first-order theory of constraints is decidable. We can define $\rho^{\#} \models^{l} c$ iff $\models \forall((\bigwedge_{x \in fv(c)} \rho^{\#}_x) \supset c)$ and $\rho^{\#} \models^{u} c$ iff $\models \exists((\bigwedge_{x \in fv(c)} \rho^{\#}_x) \wedge c)$. These approximations are safe and optimal. However, this case seems to be quite hard to happen in practice. Dual approximations of constraints may be found in general from dual approximations of boolean expressions $[\![be]\!]^{\#}$ in the obvious way by taking $[\![c_1 \wedge c_2]\!]^{\#} = [\![c_1]\!]^{\#} \wedge [\![c_2]\!]^{\#}$ and so on.

11 Related Works

The combination of abstract interpretation and model checking has been the topic of intensive research in the last few years. Safeness of abstract models has been investigated first in the universal fragment of branching-time logics [2,3,1,20]. The extension to both universal and existential properties pointed out some basic difficulties that have been solved by the idea of dual abstract transition relations *constrained* and *free* [11,10,14,4]. The same result has been found by Kelb [14] which defines safeness of abstract model checking in a classical way as safeness of the μ-calculus semantics rather than as safeness of the underlying abstract model. In order to deal with not-monotonic negation the use of dual approximations is suggested. As we have discussed the dual abstract transition relations lead equivalently to dual approximations of the next modality.

Given a correct condition of safeness the effective application of abstract model checking requires formal methods to build abstract models directly from programs without looking at the concrete model. This problem has been tackled mainly for the universal fragment of branching-time temporal logics. In particular in the framework of value-passing concurrent processes [6] proposes an abstract labelled transition system semantics for abstract closed processes obtained by value abstraction. The class of temporal properties that is preserved by the abstract models seems to be limited to the universal fragment. Schmidt [20] shows a methodology for computing a finite approximate semantics of value-passing CCS by finitely approximating the semantics over abstract environments as a regular tree. Such an approximation is based on control-abstraction and preserves universal properties only. For full μ-calculus without explicit negation a systematic method to derive a safe abstract model by symbolic execution of simple programs has been proposed by [10]. The lower approximation of the next modality (i.e. the constrained transition relation) is less precise than the one obtained in our approach thanks to $\oplus$ and $\otimes$, since it suffers of the loss of information due to alternative choices discussed in example 16.

For linear-time temporal logics safeness is simple as in the universal fragment of branching-time temporal logics. In this setting [15] proposes a method to safely approximate fairness constraints and suggests the use of deduction to establish safeness of the abstract model. [21] tries to verify a temporal formula of linear temporal logic even of infinite systems using a finite graph representation, that is an abstraction of the infinite one. The idea is that of refining the tableau of formula $\neg A$ until either a counterexample is found or the formula is proved.

As far as concern the symbolic semantics a method for representing regular processes by finite graphs has been introduced in [19]. Symbolic graphs with assignments are similar to the one obtained by means of generalization, but in general larger.

12 Conclusions

In this paper we have proposed a method to apply finite value abstractions to the model checking of μ-calculus and value-passing concurrent processes. The main contribution is the definition of a symbolic semantics of processes for representing the classical infinite labelled transition system. The proposed symbolic graph differs from the classical one [12], since classical branching of transitions is replaced by explicit relations of alternative and non-deterministic choices among transitions. Moreover, the classical infinite paths due to parameterized recursion are avoided by generalizing current recursive calls so that the graph is finite for regular processes. The symbolic graph provides the basis for realizing abstract model checking by finite value abstraction. Technically we have defined a method to obtain a safe lower approximate semantics of the full logic by interpreting symbolic transitions over abstract values. Explicit negation is treated by the combination of dual approximations as suggested by [14].

In the proposed approach the abstract model is implicitly derived from the symbolic graph with respect to abstract values. The basic approximation step concerns the next modality. Free and constrained abstract transitions are derived from symbolic transitions by approximating simple constraints expressing the conditions for the existence of transitions. We have shown that the relations of alternative and non-deterministic choices allow us to achieve in general a more precise result with respect to previous proposals [11,10]. Specific solutions for the dual approximation of constraints should be studied to further improve the precision of abstract model checking depending on the chosen domain of boolean and value expressions. However, since constraints abstraction is a common practice in data-flow analysis it seems that several well-known methods could be successfully used to this aim.

Precision of abstract model checking depends dramatically on constraints approximation and in addition on the specific value abstraction that has been chosen a priori. The main problem is that it is very difficult in general to guess a sufficiently precise abstraction for the formula to be proved so that subsequent refinements are typically necessary. For this purpose the construction at run-time of the abstract model on the symbolic graph is very useful, since it allows us to recompute the approximate semantics without reconstructing the whole new abstract model.

There are many interesting problems to study in the future. The main weakness of abstract model checking is undoubtedly the choice a priori of a proper finite abstraction. It would be important to develop formal techniques for partially exploiting the features of the formula and of the system in order to guess the abstraction. This problem leads to a very close question: there exists a class of formulas for which a precise and finite abstraction exists? Precise means that the formula either holds or do not hold in the abstract model. These results would be essential to effectively compare the power and complexity of abstract model checking with the ones of the existing deductive methods.

This paper is a revised and complete version of [18].

Acknowledgements

I would like to thank Radhia and Patrick Cousot for having me introduced to the abstract interpretation theory. I also thank Dave Schmidt for his helpful suggestions concerning the presentation of this work.

References

[1] S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property preserving simulations. In *Proceedings of CAV 92*, volume 663 of *Lecture Notes in Computer Science*, pages 260–263. Springer-Verlag, Berlin, 1992.

[2] E.M. Clarke, O. Grumberg, and D.E. Long. Model Checking and Abstraction. In *Proc. 19th Annual ACM Symp. on Principles of Programming Languages*, pages 343–354. ACM Press, 1992.

[3] E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 5(16):1512–1542, 1994.

[4] R. Cleaveland, P. Iyer, and D. Yankelevic. Optimality in Abstractions of Model Checking. In *Proceedings of SAS 95*, volume 983 of *Lecture Notes in Computer Science*, pages 51–63. Springer-Verlag, Berlin, 1995.

[5] R. Cleaveland, T. Margaria, and B. Steffen. Efficient Local Model Checking for Fragments of μ-calculus. In *Tools and Algorithms for the Constructon and Analysis of Software (TACAS 96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 107–126. Springer-Verlag, Berlin, 1996.

[6] R. Cleaveland and J. Riely. Testing based abstractions for value-based systems. In *Proceedings of CONCUR 94*, volume 836 of *Lecture Notes in Computer Science*, pages 417–432. Springer-Verlag, Berlin, 1994.

[7] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. Fourth ACM Symp. Principles of Programming Languages*, pages 238–252, 1977.

[8] P. Cousot and R. Cousot. Systematic Design of Program Analysis Frameworks. In *Proc. Sixth ACM Symp. Principles of Programming Languages*, pages 269–282, 1979.

[9] P. Cousot and R. Cousot. Abstract Interpretation and Applications to Logic Programs. *Journal of Logic Programming*, 13(2 & 3):103–179, 1992.

[10] D. Dams, R. Gerth, and O. Grumberg. Abstract Interpretation of Reactive Systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, 1997.

[11] D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving $\forall CTL^*$, $\exists CTL^*$ and CTL^*. In *Proceedings of the Working Conference on Programming Concepts, Methods and Calculi (PROCOMET)*, 1994.

[12] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138:353–389, 1995.

[13] J.R.Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and J. Hwang. Symbolic Model Checking. In *Proc. of the Fifth Annual Logic in Computer Science*, pages 428–439, 1990.

[14] P. Kelb. Model Checking and Abstraction: A framework preserving both truth and failure information. Technical report, OFFIS, Oldenburg, Germany, 1994.

[15] Y. Kesten and A. Pnueli. Modularization and Abstraction: The Keys to Practical Formal verification. In *23rd International Symp. Mathematical foundations of Computer Science, MFCS 98*, volume 1450 of *Lecture Notes in Computer Science*, pages 54–71. Springer-Verlag, Berlin, 1998.

[16] D. Kozen. Results on the Propositional mu-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[17] F. Levi. Abstract model checking of value-passing processes. In A. Bossi, editor, *International Workshop on Verification, Model Checking and Abstract Interpretation*, 1997. http://www.dsi.unive.it/ bossi/VMCAI.html.

[18] F. Levi. A Symbolic Semantics for Abstract Model Checking. In G. Levi, editor, *Fifth International Static Analysis Symposium, SAS 98*, volume 1503 of *Lecture Notes in Computer Science*, pages 134–151. Springer-Verlag, Berlin, 1998.

[19] H. Lin. Symbolic Transition Graph with Assignment. In *Proc. of CONCUR 96*, volume 1119 of *Lecture Notes in Computer Science*, pages 50–65. Springer-Verlag, Berlin, 1996.

[20] D.A. Schmidt. Abstract Interpretation of Small-Step Semantics. In *Proc. of the LOMAPS Workshop on "Analysis and Verification of Multiple-Agent Languages"*, volume 1192 of *Lecture Notes in Computer Science*, pages 76–99, 1996.

[21] Henny B. Sipma, Tomas E. Uribe, and Z. Manna. Deductive Model Checking. In *8th International Conference on Computer Aided Verification, CAV 96*, volume 1102 of *Lecture Notes in Computer Science*, pages 208–219. Springer-Verlag, Berlin, 1996.

[22] C. Stirling and D. Walker. Local Model Checking in the Modal mu-Calculus. *Theoretical Computer Science*, 89:161–177, 1991.

A Appendix

Theorem 10 Let $P \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$, where $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ and $\Omega_i = \otimes_{j_i\in\{1,n_i\}}P_{i,j_i}$, be a symbolic transition. For each environment $\rho \in Env$ there exists one and only one $i \in \{1,n\}$ such that $\rho \models c_i$ and

(1) for each $j_i \in \{1,n_i\}$, if $\theta_{i,j_i} = c?x$ then $(P,\rho) \stackrel{c?v}{\mapsto} p \equiv (P_{i,j_i}, \rho[x \to v])$, for $v \in Val$, if $\theta_{i,j_i} = \tau$ then $(P,\rho) \stackrel{\theta_{i,j_i}}{\mapsto} p \equiv (P_{i,j_i}, \rho)$, and if $\theta_{i,j_i} = c!e$ then $(P,\rho) \stackrel{c!v}{\mapsto} p \equiv (P_{i,j_i}, \rho)$, where $\mathcal{S}_v(e\rho) = v$;

(2) for each $(P,\rho) \stackrel{a}{\mapsto} p$ there exists $j_i \in \{1,n_i\}$ such that, if $a = c?v$ then $\theta_{i,j_i} = c?x$ and $(P_{i,j_i}, \rho[x \to v]) \equiv p$, if $a = \tau$ then $\theta_{i,j_i} = a$ and $(P_{i,j_i}, \rho) \equiv p$, and if $a = c!v$ then $\theta_{i,j_i} = c!e$ and $(P_{i,j_i}, \rho) \equiv p$, where $\mathcal{S}_v(e\rho) = v$.

PROOF. By induction on the structure of the process.

(1) Basic process. Trivial.

(2) Choice. Let $P_i \stackrel{\otimes_{j_i\in\{1,n_i\}}\Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i\in\{1,n_i\}}\Omega_{i,j_i}$ such that $\Theta_{i,j_i} = (c_{i,j_i}, \theta_{i,j_i,1} \oplus \ldots \oplus \theta_{i,j_i,k_{j_i}})$ and $\Omega_{i,j_i} = P_{i,j_i,1} \oplus \ldots \oplus P_{i,j_i,k_{j_i}}$, for $i \in \{1,2\}$, $j_i \in \{1,n_i\}$.
For each $\rho \in Env$, there exists one and only one $j_i \in \{1,n_i\}$, for each $i \in \{1,2\}$, such that $\rho \models c_{i,j_i}$. Therefore, there exists one and only pair $j_1 \in \{1,n_1\}, j_2 \in \{1,n_2\}$ such that $\rho \models c_{1,j_1} \wedge c_{2,j_2}$. Let $\Theta_{1,j_1} + \Theta_{2,j_2} = (c_{1,j_1} \wedge c_{2,j_2}, \oplus_{i\in\{1,2\},h_i\in\{1,k_{j_i}\}}\tilde{\theta}_{i,j_i,h_i})$ and $\Omega_{1,j_1} + \Omega_{2,j_2} = \oplus_{i\in\{1,2\},h_i\in\{1,k_{j_i}\}}\tilde{P}_{i,j_i,h_i}$ such that $\rho \models c_{1,j_1} \wedge c_{2,j_2}$. Properties 1) and 2) follow by induction hypothesis and by definition 5.

(3) Conditional. Let $P_i \stackrel{\otimes_{j_i\in\{1,n_i\}}\Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i\in\{1,n_i\}}\Omega_{i,j_i}$ such that $\Theta_{i,j_i} = (c_{i,j_i}, \theta_{i,j_i,1} \oplus \ldots \oplus \theta_{i,j_i,k_{j_i}})$ and $\Omega_{i,j_i} = P_{i,j_i,1} \oplus \ldots \oplus P_{i,j_i,k_{j_i}}$, for $i \in \{1,2\}$, $j_i \in \{1,n_i\}$.
For each $\rho \in Env$ there exists one and only one $j_i \in \{1,n_i\}$, for each $i \in \{1,2\}$, such that $\rho \models c_{i,j_i}$. For each $\rho \in Env$, either $\rho \models be$ or $\rho \models \neg be$. Suppose $\rho \models be$, then there exists one and only one $j_1 \in \{1,n_1\}$ such that $\rho \models be \wedge c_{1,j_1}$. Let $\Theta^{be}_{1,j_1} = (be \wedge c_{1,j_1}, \tilde{\theta}_{1,j_1,1} \oplus \ldots \oplus \tilde{\theta}_{1,j_1,k_{j_1}})$ and $\Omega^{be}_{1,j_1} = \tilde{P}_{1,j_1,1} \oplus \ldots \oplus \tilde{P}_{1,j_1,k_{j_1}}$, such that $\rho \models be \wedge c_{1,j_1}$.Properties 1) and 2) follow by induction hypothesis and by definition 6. The case of $\rho \models \neg be$ is analogous.

(4) Recursion and restriction. Trivial by induction and rules.

(5) Parallel Composition. Let $P_i \stackrel{\otimes_{j_i\in\{1,n_i\}}\Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i\in\{1,n_i\}}\Omega_{i,j_i}$ such that $\Theta_{i,j_i} = (c_{i,j_i}, \theta_{i,j_i,1} \oplus \ldots \oplus \theta_{i,j_i,k_{j_i}})$ and $\Omega_{i,j_i} = P_{i,j_i,1} \oplus \ldots \oplus P_{i,j_i,k_{j_i}}$, for $i \in \{1,2\}$, $j_i \in \{1,n_i\}$.
For each $\rho \in Env$, there exists one and only one $j_i \in \{1,n_i\}$, for each $i \in \{1,2\}$, such that $\rho \models c_{i,j_i}$. Therefore, there exists one and only $j_1 \in \{1,n_1\}, j_2 \in \{1,n_2\}$ such that $\rho \models c_{1,j_1} \wedge c_{2,j_2}$. Let $\Theta_1 \times \Theta_2 = (c_{1,j_1} \wedge c_{2,j_2}, \oplus_{i\in\{1,2\},h_i\in\{1,k_{j_i}\}}\theta_{1,j_1,h_1} \hat{\times} \theta_{2,j_2,h_2} \oplus_{i\in\{1,2\},h_i\in\{1,k_{j_i}\}} \tilde{\theta}_{1,j_i,h_i})$

and $\Omega_1 \times \Omega_2 = \oplus_{i\in\{1,2\},h_i\in\{1,k_{j_i}\}} P_{1,j_1,h_1} \hat{\times} P_{2,j_2,h_2} \oplus_{i\in\{1,2\},h_i\in\{1,k_{j_i}\}} \tilde{P}_{1,j_i,h_i}$ such that $\rho \models c_{1,j_1} \wedge c_{2,j_2}$.

1) For each $h_1 \in \{1, h_{j_1}\}, h_2 \in \{1, h_{j_2}\}$ such that $\theta_{1,j_1,h_1} \hat{\times} \theta_{2,j_2,h_2} = \tau$, by induction hypothesis $(P_1, \rho) \overset{a?v}{\mapsto} (P_{1,j_1,h_1}, \rho[x \to v])$ and $(P_2, \rho) \overset{a!v}{\mapsto} (P_{2,j_2,h_2}, \rho)$, where $\theta_{1,j_1,h_1} = c?x$, $\theta_{2,j_2,h_2} = c!e$ and $\mathcal{S}_v(e\rho) = v$. By rule $\times_2$, $(P_1 \times P_2, \rho) \overset{\tau}{\mapsto} (P_{1,j_1,h_1}, \rho[x \to v]) \times (P_{2,j_2,h_2}, \rho)$. By lemma 9 $(P_{1,j_1,h_1}, \rho[x \to v]) \times (P_{2,j_2,h_2}, \rho) \equiv (P_{1,j_1,h_1}[v/x], \rho) \times (P_{2,j_2,h_2}, \rho)$. Obviously, $(P_{1,j_1,h_1}[v/x], \rho) \times (P_{2,j_2,h_2}, \rho) \equiv .(P_{1,j_1,h_1}[e/x] \times P_{2,j_2,h_2}, \rho)$

For each $h_i \in \{1, k_{j_i}\}$, the transitions corresponding to $\tilde{\theta}_{1,j_i,h_i}$ trivially follows by induction hypothesis and rule $\times_1$.

2) For each $(P_1 \times P_2, \rho) \overset{a}{\mapsto} P$, there are two cases: either rule $\times_1$ or $\times_2$ has been applied. If rule $\times_1$ has been applied, then the thesis follows from induction hypothesis.

If rule $\times_2$ has been applied, then $(P_1, \rho) \overset{a?v}{\mapsto} p_1 \equiv (P_{1,j_1,h_1}, \rho[x \to v])$ and $(P_2, \rho) \overset{a!v}{\mapsto} p_2 \equiv (P_{2,j_2,h_2}, \rho)$. By induction hypothesis, there exist $h_1 \in \{1, k_{j_1}\}$ and $h_2 \in \{1, k_{j_2}\}$ such that $\theta_{1,j_1,h_1} = c?x$ and $\theta_{2,j_2,h_2} = c!e$, where $\mathcal{S}_v(e\rho) = v$. Therefore, $\theta_{1,j_1,h_1} \hat{\times} \theta_{2,j_2,h_2} = \tau$ and $P_{1,j_1,h_1} \hat{\times} P_{2,j_2,h_2} = P_{1,j_1,h_1}[e/x] \times P_{2,j_2,h_2}$. Obviously, $(P_{1,j_1,h_1}[e/x] \times P_{2,j_2,h_2}, \rho) \equiv (P_{1,j_1,h_1}[v/x] \times P_{2,j_2,h_2}, \rho) \equiv p_1 \times p_2$. $\square$

Theorem 22 For each closed formula A, both $p \in [\![A]\!]$ implies $(P, \rho) \in [\![A]\!]^S$ for each $P \in GP^*$ and $\rho \in Env$ such that $p \equiv (P, \rho)$, and $(P, \rho) \in [\![A]\!]^S$ implies $p \in [\![A]\!]$ for each $p \in P^*$ such that $p \equiv (P, \rho)$.

PROOF. By induction on the structure of the formula. The only interesting case is that of the the next modality.

(1) Let $(P, \rho) \in [\![K]\!]^S(S)$, where $S = [\![A]\!]^S$ for some formula A. By definition 21 there are two cases: either $\rho \models \psi(P, K, S)$ and $K \in \{c!V, \tau\}$ or $\rho[x \to v] \models \psi(P, K, S)$ and $K = c?V$, $v \in V$. Suppose $K = \tau$. Since $\rho \models \psi(P, K, S)$ there exists $i \in \{1, n\}$ and $j_i \in \{1, n_i\}$ such that $\rho \models c_i$ and $\rho \models \theta_{i,j_i} \preceq K \wedge P_{i,j_i} \preceq S$. By theorem 10 $(P, \rho) \overset{\tau}{\mapsto} (P_{i,j_i}, \rho)$ and, for each $p \in P^*$ such that $p \equiv (P, \rho)$, then $p \overset{\tau}{\mapsto} p'$ and $p' \equiv (P_{i,j_i}, \rho)$. By proposition 14 $(P_{i,j_i}, \rho) \equiv (GP, \rho')$ for some $(GP, \rho') \in S$. Therefore, $p' \equiv (GP, \rho')$ and by induction hypothesis $p' \in P^*$ implies $p' \in [\![A]\!]$. Thus, $p' \in [\![\tau]\!]([\![A]\!])$. The other cases are similar. In the case of $K = c?V$, $\rho[x \to v] \models \psi(P, K, S)$ implies $\rho \models c_i$ by lemma 9.

(2) Let $p \in [\![K]\!]([\![A]\!])$. By definition there exists $p \overset{a}{\mapsto} p'$ where $a \in K$ and $p' \in [\![A]\!]$. Suppose $a = \tau$. By theorem 17 there exists $P \in GP^*$ and $\rho \in Env$ such that $(P, \rho) \equiv p$. By theorem 10 there exist $i \in \{1, n\}$ and $j_i \in \{1, n_i\}$ such that $\rho \models c_i \wedge \theta_{i,j_i} \preceq K$ and $(P_{i,j_i}, \rho) \equiv p'$. Let $GP_{i,j_i} \in GP^*$ such that $GP_{i,j_i} \Longrightarrow_\sigma P_{i,j_i}$. By proposition 14 $p' \equiv (P_{i,j_i}, \rho) \equiv (GP_{i,j_i}, \rho \bigtriangleup \sigma)$. Since $p' \in [\![A]\!]$, by induction hypothesis $(GP_{i,j_i}, \rho) \in [\![A]\!]^S$ so that $\rho \models P_{i,j_i} \in S$.

Therefore, $\rho \models \psi(P, K, S)$. The other cases are similar. In the case of $K = c?V$, $\rho[x \to v] \models \psi(P, K, S)$ note that $\rho[x \to v] \models c_i$ by lemma 9. $\square$