A Symbolic Semantics for Abstract Model Checking *

Francesca Levi

Dipartimento di Informatica, Universitá di Pisa
and
LIX, École Polytechnique
levifran@di.unipi.it

Abstract. We present a finite symbolic semantics of value-passing concurrent processes, that can be suitably interpreted over abstract values to compute a lower approximate semantics of full μ-calculus. The main feature of the semantics is that classical branching is replaced by explicit relations of non-deterministic and alternative choices among transitions. A combination of safe upper and lower approximations of the basic operators of the logic is used to handle negation. The relations of non-deterministic and alternative choices turn out to be very useful for the dual approximations of the existential next modality.

Key words Model checking, μ-calculus, abstract interpretation.

1 Introduction

Model Checking is a very successful technique for the automatic verification of temporal properties of reactive and concurrent systems, but it is only applicable to finite-state systems. Over the past few years, abstract interpretation has been widely applied to handle large as well as infinite systems with model checking [3, 1, 8, 10, 12, 4, 16, 9, 14]. Abstract interpretation [6, 7] was originally conceived in the framework of data-flow analysis for designing approximate semantics of programs and relies on the idea of obtaining an approximate semantics from the standard one by substituting the concrete domain of computation and its basic operations with an abstract domain and corresponding abstract operations. The typical approach consists of constructing an *abstract model* over a chosen set of abstract states that can be used in model checking instead of the concrete one. To this aim the abstract model has to be safe, namely the formulas satisfied by the abstract model have to hold in the concrete one. For branching time logics the definition of a safe abstract transition relation among abstract states presents some basic difficulties and a single abstract transition relation cannot preserve both the existential and the universal next modality. Several authors [10, 9, 4]

* This work has been partially supported by the HCM project ABILE (ERBCHRXCT940624).

propose to adopt two different abstract transition relations: a *free* transition relation for computing the universal next modality, and a *constrained* transition relation for computing the existential next modality. Safeness of the free transition relation is ensured, if every concrete transition induces a free transition among the corresponding abstract states. In contrast, a constrained transition between abstract states is safe only if all the corresponding concrete transitions exist. Given this notion of safeness it turns out to be very difficult to effectively compute a sufficiently precise safe abstract model without constructing the concrete one.

In this paper we propose a method for applying abstract interpretation to the μ-calculus model checking of value-passing concurrent processes. The main contribution is the definition of a symbolic semantics of processes in the style of [11], whose main feature is that explicit relations of non-determinism and alternative choice among transitions replace classical branching. Moreover, a finite graph for regular processes is achieved by avoiding the infinite paths of [11] due to parameterized recursion. Model checking of μ-calculus can be suitably performed by interpreting the obtained symbolic graph over concrete environments assigning concrete values to variables. However, since processes are capable of exchanging values taken from a typically infinite set, the fixpoint computation of μ-calculus semantics is not effectively computable. We define a technique to compute a lower approximation of the μ-calculus semantics by interpreting the symbolic graph over abstract environments on a given (finite) set of abstract values. Safeness of the lower approximation ensures indeed the preservation of any property. Following [12] the lower approximation is achieved by combining dual safe upper and lower abstract functions corresponding to all logical connectives except negation. The critical case is undoubtedly that of the next modality, where safe constrained and free transition relations among abstract processes have to be considered. We show that explicit non-deterministic and alternative choices between transitions allow us to avoid some typical problems due to abstract branching so that a more precise lower approximation of the next modality in particular is achieved with respect to previous proposals [9]. Finally, we discuss the basic problems that typically lead to miss optimality in the approximations of the next modalities.

The paper is organized as follows. Section 2 presents value-passing concurrent processes, μ-calculus and concrete model checking. Section 3 summarizes the basic concepts of abstract interpretation. The symbolic graph is described in section 4 and the corresponding model checking algorithm is shown in section 5. Section 6 presents abstract model checking and section 7 discuss optimality of abstract model checking.

2 Concrete Model Checking

We consider a value-passing version of CCS. Let Val a set of values (possibly infinite), $Chan$ a set of channels and Var a set of variables. Moreover, let $Bexp$ and $Vexp$ be sets of boolean and values expressions. 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 $a \in \{c!e, c?x, \tau \mid c \in Chan, e \in Vexp, x \in Var\}$, $L \subseteq Chan$, and $be \in Bexp$ is a boolean expression and $e_i \in Vexp$ are expressions over values. Process $c?x.p$ will receive a value v on the channel c and then behaves as $p[v/x]$, where $p[v/x]$ denotes the standard substitution of v for all free occurrences of x. Process $c!e.p$ will send the value of the expression e and then behaves as p. The operator $+$ represents choice, while $\times$ represents parallel composition. Process $be \bigtriangledown p_1, p_2$ behaves as p_1 if the value of be is true, and as p_2 otherwise. The operator $\setminus L$ is the standard restriction for a set of channels L. Finally, $P(x_1, \ldots, x_n)$ is a process constant, which has an associated definition $P(x_1, \ldots, x_n) \equiv p$. We assume the usual definitions of free variables $fv(p)$ and bound variables $bv(p)$ of processes. A process p is closed iff $fv(p) = \emptyset$. In the following, we denote by capital letters $T, P \ldots$ open processes. For recursive processes $P(x_1, \ldots, x_n) \equiv p$ we assume $fv(p) \subseteq \{x_1, \ldots, x_n\}$ and we assume the body to be guarded. The concrete semantics of processes is defined in table 1 of the appendix as a labelled transition system $LTS(p) = (P^*, \overset{a}{\mapsto})$ with actions $a \in Act = \{\tau, c?v, c!v \mid c \in Chan, v \in Val\}$. Two semantic functions $\mathcal{S}_v : Vexp \to Val$ and $\mathcal{S}_b : Bexp \to \{tt, ff\}$ for the evaluation of expressions are used. For $a \in Act$, we define $chan(\tau) = \emptyset$, $chan(c?v) = chan(c!v) = \{c\}$. Moreover, we denote by $\bar{a}$ the symmetric action of a, namely $c!v$ for $c?v$ and $c?v$ for $c!v$. Note that, if the set of values is infinite, the labelled transition system is infinite and infinitely branching.

For expressing temporal properties of processes we consider a simple extension of propositional μ-calculus [13]. Let Act 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 $K \in \{\tau, c?V, c!V \mid c \in Chan, V \subseteq Val\}$. We assume that each variable X occurs positively in formulas. The modality $< K >$ subsumes the classical existential next modality $< a >$ ranging over actions $a \in Act$ and 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 fixpoint and the dual operator of greatest fixpoint is equivalent to $\nu X.A \equiv \neg \mu X.A[\neg X/X]$. Note that formulas with K infinite are needed for subsuming the logics CTL and CTL^*, since Act can be infinite. For instance, the classical liveness property $\forall F A$ can be expressed as $\mu X.A \vee [\tau]X \bigvee_{c \in Chan}[c?Val]X \bigvee_{c \in Chan}[c!sVal]X$.

Traditional global model checking corresponds to compute the semantics of the formula $\| A \|$ on the concrete labelled transition system $LTS(p) = (P^*, \overset{a}{\mapsto})$. Let $\delta : VAR \to \mathcal{P}(P^*)$ be a valuation assigning subsets of P^* to logical variables. The semantics of an open formula A with respect to δ is defined as:

$$\| X \|_\delta = \delta(X) \qquad \| \neg A \|_\delta = P \setminus \| A \|_\delta$$
$$\| A_0 \wedge A_1 \|_\delta = \| A_0 \|_\delta \cap \| A_1 \|_\delta \qquad \| \mu X.A \|_\delta = \mu V.(\| A \|_{\delta[V/X]})$$
$$\| < K > A \|_\delta = \cup_{a \in K} \| < a > \| (\| A \|_\delta)$$

where $\delta[V/X]$ stands for the valuation δ', which agrees with δ except that $\delta'(V) = \delta(X)$. The *next modality function* $\|< a >\|: \mathcal{P}(P^*) \to \mathcal{P}(P^*)$ is given by $\|< a >\| (S) = \{p \in P \mid \exists p \overset{a}{\mapsto} p', p' \in S\}$.

Therefore, $p \in\| A \|$ $(p \models A)$ iff $p \in\| A \|_{\delta_0}$, where δ_0 is the empty evaluation.

3 Abstract Interpretation Theory

In this section we briefly recall the basic ideas of abstract interpretation, we refer the reader to [6, 7] for more details. 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.

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

Intuitively, 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 the concretization process introduce no loss of information. Let $\mathcal{S}(P)$ be the semantics of a program P computed as the least fixpoint of a semantic function F over the concrete domain $(C, \leq)$. The goal is that of computing an approximate semantics $\mathcal{S}^{\#}(P)$ over $(A, \leq^{\#})$, that is *safe* $\alpha(\mathcal{S}(P)) \leq^{\#} \mathcal{S}^{\#}(P)$. The main result is that a safe approximate semantics $\mathcal{S}^{\#}(P)$ can be computed as the least fixpoint of a safe approximate semantic function $F^{\#}$ over $(A, \leq^{\#})$, such that $\alpha(F(c)) \leq^{\#} F^{\#}(\alpha(c))$, for each $c \in C$. Moreover, it has been shown there there exists always a best approximate semantic function $F^{\#}$ (*optimal*), when $\alpha(F(c)) = F^{\#}(\alpha(c))$. In the Galois insertion case this is equivalent to $\alpha(F(\gamma(a)) = F^{\#}(a)$, for each $a \in A$.

4 The Symbolic Graph

In this section we define the symbolic graph of processes. Symbolic semantics as introduced in [11] relies on the idea of using *symbolic actions* instead of concrete actions. For instance, transitions modeling input are represented by a single transition $c?x.p \overset{c?x}{\mapsto} p$. To handle conditional open processes symbolic actions depend in addition on boolean guards. Thus, a symbolic transition $T \overset{(c,\theta)}{\mapsto} T'$ represents all concrete transitions with action a corresponding to symbolic action θ for the assignments to the free variables of T such that guard c is satisfied. Our definition of the symbolic graph differs in some aspects from the classical one. First, classical branching is replaced by explicit relations of non-deterministic $(\oplus)$ and alternative choices $(\otimes)$ among transitions. Moreover, a method to avoid infinite applications of the standard rule for recursion is proposed. This way a finitely branching and finite graph for regular processes is obtained.

We use the standard notions of substitutions and environments. A *substitution* is a partial function $\sigma : Var \rightarrow VExp$ and a *simple substitution* is an injective $\sigma : Var \rightarrow Var$. We denote by Sub the set of simple substitutions. For a substitution σ we denote by $tar(\sigma)$ and $dom(\sigma)$ its target and source, respectively. An *environment* is a total function $\rho : Var \rightarrow Val$ and Env is the set of environments. For an environment ρ we denote by $\rho[x \rightarrow v]$ the environment that agrees with ρ except that the value assigned to variable x is v. Let T be an open process and $x \in Var$. We say that x is *fresh* in T iff $x \notin fv(T) \cup bv(T)$. Moreover, we say that a term T is free for a simple substitution σ with $dom(\sigma) \subseteq fv(T)$, iff for each $x \in fv(tar(\sigma))$, $x \notin bv(T) \cup (fv(T) \setminus tar(\sigma))$. Let T be an open process and ρ an environment. We denote by (T, ρ) the closed process $T\rho$, that is obtained by substituting $\rho(x)$ to variable x for each $x \in fv(T)$.

Let $\mathcal{C}$ be the set of *constraints* with $c ::= be \mid e = e \mid e \in V \mid true \mid \neg c \mid c \wedge c$, where $e \in Vexp$, $be \in Bexp$ and $V \subseteq Val$. We denote by $vars(c)$ the variables occurring in constraint c. For $c \in \mathcal{C}$ we consider the semantic function $\parallel c \parallel : \mathcal{P}(Env) \rightarrow \mathcal{P}(Env)$ obtained in the trivial way by $\parallel be \parallel = \{\rho \in Env \mid \mathcal{S}_b(be\rho) = tt\}$, $\parallel e_1 = e_2 \parallel = \{\rho \in Env \mid \mathcal{S}_v(e_1\rho) = \mathcal{S}_v(e_2\rho)\}$ and $\parallel e \in V \parallel = \{\rho \in Env \mid \mathcal{S}_v(e\rho) \in V\}$. We use the notation $\rho \models c$ for $\rho \in \parallel c \parallel$.

Let the *symbolic actions* be $SymAct = \{c?x, c!e, \tau \mid c \in Chan, x \in Var, e \in Vexp\}$ with $bv(c?x) = \{x\}$, $bv(c!e) = bv(\tau) = \emptyset$ and $fv(c!e) = vars(e)$, $fv(\tau) = fv(c?x) = \emptyset$.

Transitions of the symbolic semantics are $T \stackrel{\otimes_{i \in \{1,n\}} \Theta_i}{\longrightarrow} \otimes_{i \in \{1,n\}} \Omega_i$, such that for each $i \in \{1, n\}$ $(i \in \{1, \ldots, n\})$

1. $\Theta_i = (c_i, \oplus_{j_i \in \{1,n_i\}} \theta_{i,j_i})$ with $\theta_{i,j_i} \in SymAct \cup \{*\}$ and $c_i \in \mathcal{C}$;
2. $\Omega_i = \oplus_{j_i \in \{1,n_i\}} T_{i,j_i}$ where T_{i,j_i} are open processes;
3. $vars(c_i) \subseteq fv(T)$, $fv(\theta_{i,j_i}) \subseteq fv(T)$ and $fv(T_{i,j_i}) \subseteq fv(T) \cup bv(\theta_{i,j_i})$.

All possible behaviors of closed processes obtained from T are represented by a single transition $T \stackrel{\otimes_{i \in \{1,n\}} \Theta_i}{\longrightarrow} \otimes_{i \in \{1,n\}} \Omega_i$, where alternative choices are related by $\otimes$ and non-deterministic choices by $\oplus$. The idea is that for each environment ρ there exists a unique alternative c_i that is satisfied and symbolic actions θ_{i,j_i} with corresponding processes T_{i,j_i} represent the concrete transitions of (T, ρ).

Let us explain informally the construction of transitions. The complete semantic rules are shown in table 2 of the appendix.

Transitions of a basic process $a.T$ are obtained by a rule $a.T \stackrel{(true, a \oplus *)}{\longrightarrow} T \oplus a.T$, where the special action $*$ denotes idle action and is used in the parallel composition rule.

The non-deterministic choice of $+$ is reflected by the composition of transitions with $\oplus$. The rule for choice is as follows

$$\frac{\{T_i \stackrel{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{T_1 + T_2 \stackrel{\otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \Theta^{+}_{j_1,j_2}}{\longrightarrow} \otimes_{i \in \{1,2\}, j \in \{1,n_i\}} \Omega^{+}_{j_1,j_2}} +$$

where $\Theta^{+}_{j_1,j_2}$ is guarded by constraint $c_{1,j_1} \wedge c_{2,j_2}$ and where all non-deterministic choices of $\Theta_{i,j_i} = (c_{i,j_i}, \oplus_{h_i \in \{1,k_{j_i}\}} \theta_{i,j_i,h_i})$ for $i \in \{1,2\}$ are merged by $\oplus$. The resulting processes are analogously combined by $\Omega^{+}_{j_1,j_2}$. In fact, for each environment that satisfies both guards both actions of T_1 and of T_2 can be chosen.

Example 4.1. For instance, a process $T = c!x.T_1 + a!x.T_2$ is modeled by $c!x.T_1 + a!x.T_2 \stackrel{(true, c!x \oplus a!x \oplus *)}{\longrightarrow} T_1 \oplus T_2 \oplus T$.

The alternative choices of a conditional process are related by $\otimes$ through the following rule

$$\frac{\{T_i \stackrel{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{be \bigtriangledown T_1, T_2 \stackrel{\otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \bar{\Theta}_{i,j_i}}{\longrightarrow} \otimes_{i \in \{1,2\}, j \in \{1,n_i\}} \bar{\Omega}_{i,j_i}} \bigtriangledown$$

where $\bar{\Theta}_{1,j_1}$ is equivalent to Θ_{1,j_1} where the guard is additionally constrained by be, while $\bar{\Theta}_{2,j_2}$ is equivalent to Θ_{2,j_2} where the guard is additionally constrained by $\neg be$. Intuitively, transitions of $be \bigtriangledown T_1, T_2$ are either transitions of T_1, if be is satisfied, or transitions of T_2, if be is not satisfied.

Example 4.2. For $T = x > o \bigtriangledown c!x.T_1 + a!x.T_2, b!x.nil$ we have $T \stackrel{\Theta_1 \otimes \Theta_2}{\longrightarrow} \Omega_1 \otimes \Omega_2$, where $\Theta_1 = (x > 0, c!x \oplus a!x \oplus *)$, $\Theta_2 = (x \leq 0, b!x \oplus *)$, $\Omega_1 = T_1 \oplus T_2 \oplus T$ and $\Omega_2 = nil \oplus T$. In fact, for each environment ρ either $x > 0$ or $x \leq 0$ is true and the process (T, ρ) is able to perform either both $c!\rho(x)$ and $a!\rho(x)$ or $b!\rho(x)$, respectively.

The rule of parallel composition is quite complex. Since a single transition must represent all behaviors combined by the relations $\oplus$ and $\otimes$, a single rule performs at the same time synchronization and interleaving. The rule is defined as follows

$$\frac{\{T_i \stackrel{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}}{\longrightarrow} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{T_1 \times T_2 \stackrel{\otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \Theta^{\times}_{j_1,j_2}}{\longrightarrow} \otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \Omega^{\times}_{j_1,j_2}} \times$$

where $\Theta^{\times}_{j_1,j_2}$ is constrained by $c_{1,j_1} \wedge c_{2,j_2}$ and its actions are all possible combinations of actions θ_{1,j_1,h_1} and θ_{2,j_2,h_2} for $\Theta_{i,j_i} = (c_{i,j_i}, \oplus_{h_i \in \{1,k_{j_i}\}} \theta_{i,j_i,h_i})$ for $i \in \{1,2\}$. The combination of processes corresponding to the combination of actions is realized by $\Omega^{\times}_{j_1,j_2}$.

Example 4.3. Consider an open process $T_1 \times T_2$ with $T_1 = x > 0 \bigtriangledown c?x.T, a?x.T$ and $T_2 = c!y+1.T + a!y-1.T$. We have $T_1 \stackrel{(x>0, c?x \oplus *) \otimes (x \leq 0, a?x \oplus *)}{\longrightarrow} T \oplus T_1 \otimes T \oplus T_1$ and $T_2 \stackrel{(true, c!y+1 \oplus a!y-1 \oplus *)}{\longrightarrow} T \oplus T \oplus T_2$. The transition resulting from parallel composition is $T_1 \times T_2 \stackrel{\Theta_1 \otimes \Theta_2}{\longrightarrow} \Omega_1 \otimes \Omega_2$, where $\Theta_1 = (x > 0, \tau \oplus c?z \oplus c!y + 1 \oplus a!y - 1 \oplus *)$, $\Theta_2 = (x \leq 0, \tau \oplus a?z \oplus c!y + 1 \oplus a!y - 1 \oplus *)$, $\Omega_1 = T[y+1/x] \times T \oplus T[z/x] \times T_2 \oplus T_1 \times T \oplus T_1 \times T \oplus T_1 \times T_2$ and $\Omega_2 = T[y-1/x] \times T \oplus T[z/x] \times T_2 \oplus T_1 \times T \oplus T_1 \times T \oplus T_1 \times T_2$. Two alternative choices corresponding to constraints

$x > 0$ and $x \leq 0$ arise from composition of guard $x > 0$ with $true$ and of guard $x \leq 0$ with $true$, respectively. For guard $x > 0$ the non-deterministic choices are obtained by the composition of actions $c?x$ and $*$ with $c!y+1, a!y-1$ and $*$, while for guard $x \leq 0$ they are obtained by the composition of actions $a?x$ and $*$ with $c!y+1, a!y-1$ and $*$. Therefore, τ actions are obtained from the synchronization of actions $c?x$ and $c!y+1$ and of $a?x$ with $a!y-1$, respectively. The others arise from interleaving by composition with $*$. 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 $y-1$ is received by T_1. Note that in the interleaving case with a receive action the variable x must be renamed to z to avoid clash of variables with the free variables of T_2.

Recursive processes are handled by the classical rule, where formal parameters are substituted by actual parameters.

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

Unfortunately, the application of rule rec leads to an infinite graph also for regular processes, where there is no parallel composition inside the scope of recursion. The semantics of [11] suffers of the same problem.

Example 4.4. Let us consider the process $P(x) \equiv c!x.(P(x+1) + P(x-1))$. Since the recursive process is unfolded infinitely times with a different argument an infinite graph arise.

$$\begin{array}{l}
P(x) \stackrel{(true, c!x \oplus *)}{\longrightarrow} P(x+1) + P(x-1) \oplus P(x) \\
P(x+1) + P(x-1) \stackrel{(true, c!x+1 \oplus *)}{\longrightarrow} P(x+1+1) + P(x+1-1) \oplus P(x+1) \\
P(x+1) + P(x-1) \stackrel{(true, c!x-1 \oplus *)}{\longrightarrow} P(x-1+1) + P(x-1-1) \oplus P(x-1) \\
\ldots
\end{array}$$

This problem can be solved by replacing in the graph transitions of $P(\bar{e})$ by transitions of a process $P(\bar{x})$ for fresh variables $\bar{x}$. The semantics of $P(\bar{e})$ can naturally be obtained by the semantics of $P(\bar{x})$ by instantiating parameters $\bar{x}$ to the actual values corresponding to the evaluation of the expressions $\bar{e}$.

Let *general processes* $\mathcal{GP}$ have the following syntax

$$GP ::= nil \mid a.T \mid P(\bar{x}) \mid GP_1 + GP_2 \mid be \bigtriangledown GP_1, GP_2 \mid GP_1 \times GP_2 \mid GP \setminus L$$

where $\bar{x}$ is a tuple of distinct variables and $T \in Proc$ is a process.

Note that $a.T$ is a general process for any process T, since there are no current recursive calls. For general processes $GP \in \mathcal{GP}$, the *recursion variables* $rv(GP)$ are defined as $rv(a.T) = rv(nil) = \emptyset$, $rv(P(\bar{x})) = \{\bar{x}\}$, $rv(GP_1 + GP_2) = rv(GP_1 \times GP_2) = rv(be \bigtriangledown GP_1, GP_2) = rv(GP_1) \cup rv(GP_2)$ and $rv(GP \setminus L) = rv(GP)$.

Our aim is that of finding a general process GP that can be used instead of T in the graph. For this purpose we define *most general terms*.

Definition 4.5. Let T be a process. We define *most general terms* $\Pi(T)$:

- if $T = nil$ or $T = a.T_1$, $\Pi(T) = \{T\}$;
- for $GP_i \in \Pi(T_i)$ such that, $rv(GP_1) \cap fv(GP_2) = \emptyset$, $rv(GP_1) \cap bv(GP_2) = \emptyset$ and vice-versa, then $GP_1 + GP_2 \in \Pi(T_1 + T_2)$, $GP_1 \times GP_2 \in \Pi(T_1 \times T_2)$ and $be \bigtriangledown GP_1, GP_2 \in \Pi(be \bigtriangledown T_1, T_2)$;
- if $P(\bar{z}) \equiv T$ and T is free for $[\bar{x}/\bar{z}]$, then $P(\bar{x}) \in \Pi(P(\bar{e}))$.

Most general terms are general processes obtained by introducing fresh and distinct variables in current recursive calls. The following property is satisfied.

Proposition 4.6. *Let T be a process. For each $GP \in \Pi(T)$, there exists a substitution σ with $dom(\sigma) = rv(GP)$ such that $GP\sigma = T$.*

The substitution σ assigns actual parameters of T to formal parameters of GP. For $\rho \in Env$, let $gen_{T,GP} : Env \to Env$, such that $\rho(x) = gen_{T,GP}(\rho)(x)$, for $x \notin rv(GP)$, and $\rho(x) = \mathcal{S}_v(\sigma(x)\rho)$, otherwise. In the environment $gen_{GP,T}(\rho)$ formal parameters of GP are instantiated to the values of the actual parameters provided by σ. The result is that the closed process $(GP, gen_{T,GP}(\rho))$ is equivalent (bisimilar) to (T, ρ). For $p_1, p_2 \in Proc$ be processes, we say that $p_1 \equiv p_2$ iff for each $a \in Act$, for each $p_1 \xrightarrow{a} p_1'$ there exists $p_2 \xrightarrow{a} p_2'$ and $p_1' \equiv p_2'$ and vice-versa.

Proposition 4.7. *Let T be a process and $\rho \in Env$. For each $GP \in \Pi(T)$, $(GP, gen_{T,GP}(\rho)) \equiv (T, \rho)$.*

By proposition 4.7 the symbolic graph where transitions of T are replaced by transitions of GP correctly models the behavior of processes.

Definition 4.8. Let T be a process. We define $\mathcal{SG}(T) = (GP^*, T^*, \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$ with $GP^* \subseteq \mathcal{GP}$, $T^* \subseteq Proc$ and transitions are $GP \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ for each $GP \in GP^*$, where

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

Example 4.9. Consider for instance the process $P(1)$, where $P(x)$ is defined in example 4.4. Since $P(z) + P(w) \in \Pi(P(x + 1) + P(x - 1))$ then $\mathcal{SG}(P(1))$ is finite:

$$P(x) \stackrel{(true,c!x\oplus *)}{\longrightarrow} P(x+1) + P(x-1) \oplus P(x)$$
$$P(z) + P(w) \stackrel{(true,c!z\oplus c!w\oplus *)}{\longrightarrow} P(z+1) + P(z-1) \oplus P(w+1) + P(w-1) \oplus P(z) + P(w)$$

This graph correctly describes the behavior of $P(1)$. For instance, the concrete computation $P(1) \stackrel{c!1}{\mapsto} P(1+1) + P(1-1) \stackrel{c!2}{\mapsto} P(1+1+1) + P(1+1-1)\ldots$ is simulated by $(P(x), \rho_1) \xrightarrow{c!1} (P(z) + P(w), \rho_2) \xrightarrow{c!2} \ldots$, where $\rho_1(x) = 1$ and

$\rho_2(z) = 2$, $\rho_2(w) = 0$. Environment $\rho_2 = gen_{T,GP}(\rho_1)$ assigns to parameters z and w the result of the evaluation of expressions $x+1$ and $x-1$ with respect to ρ_1. Note that in $P(z) + P(w)$ distinct fresh variables are used to model all recursive calls $P(e_1) + P(e_2)$, where e_1 may be different from e_2.

The equivalence is formally stated by the following theorem. Let $K \in \{\tau, c!V, c?V\}$ and $\theta \in SymAct$. We define the constraint $\theta \in K \in \mathcal{C}$ as $\theta \in K \equiv true$, for $K = \theta = \tau$, $\theta \in K \equiv true$, for $\theta = c?x$ and $K = c?V$, $\theta \in K \equiv e \in V$, for $\theta = c!e$ and $K = c!V$, and $\theta \in K \equiv false$, otherwise. For $K = a$ we denote by $a = \theta$ the constraint $\theta \in a$.

Theorem 4.10. *1. For each symbolic transition* $GP \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$, *where* $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ *and* $\Omega_i = \otimes_{j_i\in\{1,n_i\}}T_{i,j_i}$ *and for each* $\rho \in Env$, *there exists one and only one* $i \in \{1,n\}$, *such that* $\rho \models c_i$ *and, for each* $j_i \in \{1, n_i\}$, *there exists* $a \in Act$ *with* $\rho \models a = \theta_{i,j_i}$ *such that,* $(GP, \rho) \overset{c?v}{\mapsto} (T_{i,j_i}, \rho[x \to v])$ *for* $\theta_{i,j_i} = c?x$ *and* $v \in Val$, *and* $(T, \rho) \overset{a}{\mapsto} (T_{i,j_i}, \rho)$, *for* $\theta_{i,j_i} \in \{\tau, c!e\}$*;*
2. for each $p \in P^*$ *there exists* $GP \in GP^*$ *and* $\rho \in Env$ *such that* $GP\rho \equiv p$ *and, for each* $p \overset{a}{\mapsto} p'$, *there exist* $i \in \{1,n\}$ *and* $j_i \in \{1, n_i\}$, *where* $GP \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ *with* $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ *and* $\Omega_i = \otimes_{j_i\in\{1,n_i\}}T_{i,j_i}$, *and* $\rho \models c_i \wedge a = \theta_{i,j_i}$ *and* $p' \equiv (T_{i,j_i}, \rho[x \to v])$, *for* $a = c?v$ *and* $\theta_{i,j_i} = c?x$, *and* $p' \equiv (T_{i,j_i}, \rho)$, *for* $a \in \{\tau, c!v\}$.

By theorem 4.10 the concrete semantics is safely represented by the symbolic one and in addition concrete non-deterministic and alternative choices are exactly composed by $\oplus$ and $\otimes$ in symbolic transitions. Since transitions are restricted to general processes, infinite applications of *rec* with different arguments are avoided and the symbolic graph is finite for regular processes.

Theorem 4.11. *Let* $p \in Proc$ *be a regular process.* $\mathcal{SG}(p)$ *is a finite graph up to renaming.*

5 *Symbolic* Model Checking

In this section we show that model checking can be realized by interpreting the symbolic graph over concrete environments. Let $(\mathcal{P}(D), \subseteq)$, with $D = \{(T, \rho) \mid T \in GP^* \text{ and } \rho \in Env\}$ for $\mathcal{SG}(p) = (GP^*, T^*, \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$. The semantics of a formula A is defined as in section 2 by replacing evaluations with symbolic evaluations $\delta : VAR \to \mathcal{P}(D)$ and by taking the following function for the next modality. For $K \in \{\tau, c!V, c?V\}$ and $S \in \mathcal{P}(D)$,

- $\| < K > \|^S (S) = \{(T, \rho) \mid T \overset{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$, with $\Theta_i = (c_i, \oplus_{j_i\in\{1,n_i\}} \theta_{i,j_i})$ and $\Omega_i = \oplus_{j_i\in\{1,n_i\}}T_{i,j_i}$, and there exist $i \in \{1,n\}$ and $j_i \in \{1, n_i\}$, such that $\rho \models c_i \wedge \theta_{i,j_i} \in K$ and $(T_{i,j}, \rho) \in_{\equiv} S$, for $\theta_{i,j_i} \in \{\tau, c!e\}$, and $(T_{i,j_i}, \rho[x \to v]) \in_{\equiv} S$, for some $v \in V$ for $K = c?V$ and $\theta_{i,j_i} = c?x\}$.

where $(T,\rho) \in_{\equiv} S$ if $(GP, gen_{T,GP}(\rho)) \in S$ for some $GP \in \Pi(T)$.

The definition of $\|< a >\|^S$ is based on the observation that process (T,ρ) is able to perform an action a, if the environment satisfies a guard c_i for which there exists a symbolic action θ_{i,j_i} corresponding to a. Moreover, the resulting process must be in S. However, since S is a set of general processes and T_{i,j_i} is not a general process we look for an equivalent process $(GP, gen_{T_{i,j_i},GP}(\rho)) \in S$.

Theorem 5.1. *For each closed formula A, $\| A \|^S \cap P^* = \| A \|$.*

6 Abstract Model Checking

In this section we define the approximate semantics of the *collecting semantics* $\| A \|^S$ on the abstract domain obtained by replacing concrete environments with abstract environments on an abstract values domain.

Let (α_v, γ_v) be a Galois insertion between the complete lattices $(\mathcal{P}(Val), \subseteq)$ and $(\mathcal{P}(Val^{\#}), \subseteq)$ of concrete and abstract values. We consider the set of *abstract environments* $Env^{\#} = \{\rho^{\#} \mid \rho^{\#} : Var \to Val^{\#}\}$ and the set of *abstract processes* $D^{\#} = \{(T, \rho^{\#}) \mid T \in GP^* \text{ and } \rho^{\#} \in Env^{\#}\}$ for $\mathcal{SG}(p) = (GP^*, T^*, \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$. Let $\gamma_e : \mathcal{P}(Env^{\#}) \to \mathcal{P}(Env)$ and $\gamma : \mathcal{P}(D^{\#}) \to \mathcal{P}(D)$ be the obvious functions induced by γ_v.

Our purpose is that of computing a lower approximation of the semantics $\| A \|^l$, such that $\gamma(\| A \|^l) \subseteq \| A \|^S$. Given the relation between concrete and abstract domain an approximate semantics can be obtained by replacing concrete functions with corresponding safe abstract functions. However, the operator of negation is not monotonic. Kelb [12] argues that a lower approximation of the full logic with negation can be obtained by combining duals approximations for formulas without negation $\| A \|^l$ and $\| A \|^u$. In fact, $\| \neg A \|^l = D^{\#} \setminus \| A \|^u$ is a safe lower approximation of $\| \neg A \|^S$, since $\| A \|^S \subseteq \gamma(\| A \|^u)$. Analogously, $\| \neg A \|^u = D^{\#} \setminus \| A \|^l$ is a safe upper approximation of $\| \neg A \|^S$. Thus, the problem is reduced to the definition of safe dual approximations for all the logical operators except negation. The safeness of the dual approximations is formally defined with respect to the following framework.

Proposition 6.1. *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 $(\mathcal{P}(D^{\#}), \supseteq)$.*

We show that both the upper and the lower approximation can be computed over the symbolic graph with respect to abstract environments. We exploit the following constraints based on relations $\oplus$ and $\otimes$. The symbol $\vee$ denotes with an abuse of notation the equivalent constraint.

Definition 6.2. Let $T \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$, with $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}}\theta_{i,j_i})$ and $\Omega_i = \otimes_{j_i\in\{1,n_i\}}T_{i,j_i}$, be a symbolic transition. For each $K \in \{\tau, c!V, c?V\}$ and $i \in \{1,n\}$, $j_i \in \{1,n_i\}$ and $I = \cup_{i\in\{1,n\}} I_i$ with $I_i \subseteq \{1,n_i\}$ we define

1. $FREE_{K,i,j_i} \equiv c_i \wedge \theta_{i,j_i} \in K$;
2. $CON_{K,I} \equiv \bigwedge_{i\in\{1,n\}} \neg c_i \vee (\bigvee_{j_i\in I_i} \theta_{i,j_i} \in K)$.

The abstract semantics is based on safe lower and upper approximations of the previous constraints and on dual approximations of the relation $\in_\equiv$. Note that for each $c \in \mathcal{C}$, $\| c \|^u$ and $\| c \|^l$ are safe approximations iff $\| c \| \subseteq \gamma_e(\| c \|^u)$ and $\gamma_e(\| c \|^l) \subseteq \| c \|$, respectively. Moreover, $\in^u_\equiv$ is a safe upper approximation, if $\rho \in \gamma_e(\rho^\#)$ such that $(T,\rho) \in_\equiv \gamma(S^\#)$ implies $(T,\rho^\#) \in^u_\equiv S^\#$, while $\in^l_\equiv$ is a safe lower approximation if $(T,\rho^\#) \in^l_\equiv S^\#$ only if, for all $\rho \in \gamma_e(\rho^\#)$, $(T,\rho) \in_\equiv \gamma(S^\#)$.

Let $\delta^\# : VAR \to \mathcal{P}(D^\#)$ be an abstract evaluation. We define the abstract upper and lower semantics of an open formula A with respect to $\delta^\#$ as follows.

Lower Approximation

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

where, for $S^\# \in \mathcal{P}(D^\#)$,

- $\| < K > \|^l (S^\#) = \{(T,\rho^\#) \mid T \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$, with $\Theta_i = (c_i, \otimes_{j_i\in\{1,n_i\}} \theta_{i,j_i})$ and $\Omega_i = \otimes_{j_i\in\{1,n_i\}} T_{i,j_i}$, and there exists $I = \cup_{i\in\{1,n\}} I_i$ with $I_i \subseteq \{1,n_i\}$, such that I is minimal and $\rho^\# \in \| CON_{K,I} \|^l$, and, for each $i \in \{1,n\}$ and $j_i \in I_i$, $(T_{i,j_i},\rho^\#) \in^l_\equiv S^\#$, for $\theta_{i,j_i} \in \{c!e,\tau\}$, and $(T_{i,j_i},\rho^\#[x \to \alpha_v(v)]) \in^l_\equiv S^\#$, for $v \in V$, $K = c?V$ and $\theta_{i,j_i} = c?x\}$.

Upper Approximation

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

where, for $S^\# \in \mathcal{P}(D^\#)$,

- $\| < K > \|^u (S^\#) = \{(T,\rho^\#) \mid T \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow} \otimes_{i\in\{1,n\}}\Omega_i$ with $\Theta_i = (c_i, \oplus_{j_i\in\{1,n_i\}} \theta_{i,j_i})$ and $\Omega_i = \oplus_{j_i\in\{1,n_i\}} T_{i,j_i}$, and there exist $i \in \{1,n\}$ and $j_i \in \{1,n_i\}$ such that $\rho^\# \in \| FREE_{K,i,j_i} \|^u$ and $(T_{i,j_i},\rho^\#) \in^u_\equiv S^\#$, for $\theta_{i,j_i} \in \{\tau, c!e\}$, and $(T_{i,j_i},\rho^\#[x \to \alpha_v(v)]) \in^u_\equiv S^\#$, for $v \in V$, $K = c?V$ and $\theta_{i,j_i} = c?x$ $\}$.

Let us explain the dual approximations of the next modality.

1. The *upper approximation* is safe, if $(T,\rho) \in \| < a > \|^S (\gamma(S^\#))$ implies $(T,\rho^\#) \in \| < a > \|^u (S^\#)$ for $\rho \in \gamma_e(\rho^\#)$. In other words, safeness requires to consider at least the abstract transitions corresponding to the concrete a-transitions (*free transition relation* [9, 4]). If constraint $FREE_{a,i,j_i} \equiv c_i \wedge \theta_{i,j_i} = a$ is safely upper approximate the previous condition is guaranteed, since $\rho \models c_i \wedge \theta_{i,j_i} = a$ implies $\rho^\# \in \| c_i \wedge \theta_{i,j_i} = a \|^u$.

2. The definition of the *lower approximation* is quite complex. In this case safeness is ensured, only if $(T, \rho^{\#}) \in \| < a > \|^l\ (S^{\#})$ implies $(T, \rho) \in \| < a > \|^S\ (\gamma(S^{\#}))$, for each $\rho \in \gamma_e(\rho^{\#})$. In other words, safeness requires to consider only the abstract transitions, for which all corresponding concrete a-transition exist (*constrained transition relation* [9, 4]). Let us consider the constraint $CON_{a,I} \equiv \bigwedge_{i \in \{1,n\}} (c_i \supset (\bigvee_{j_i \in I_i} (a = \theta_{i,j_i})))$ and its lower approximation. If $\rho^{\#} \in \| CON_{a,I} \|^l$, then for each $\rho \in \gamma_e(\rho^{\#})$, if $\rho \models c_i$ there exists an action θ_{i,j_i} with $j_i \in I_i$ corresponding to a. By theorem 4.10 for each ρ there exists exactly one c_i such that $\rho \models c_i$ so that, for each ρ there exists indeed a transition with action a.

Note that by definition of the next modality safeness for both approximations requires in addition safeness of the dual approximations of $\in_{\equiv}$ to check the resulting processes to be in $S^{\#}$.

Example 6.3. Consider a process $T = x > 0 \bigtriangledown c?x.T_1, a?x.T_2$, whose behavior is described by $T \overset{(x>0, c?x \oplus *) \otimes (x \leq 0, a?x \oplus *)}{\longrightarrow} T_1 \oplus T \otimes T_2 \oplus T$. Let the values abstraction be $\alpha_v(n) = \bullet$ and let $\rho^{\#}$ be the abstract environment with $\rho^{\#}(x) = \bullet$. For a formula $A \equiv < c?n > true$ we obtain an upper approximation $\{(T, \rho^{\#})\} = \| A \|^u$. In fact, there exists $\rho \in \gamma_e(\rho^{\#})$ such that $\rho \models x > 0 \wedge (c?x = c?n) \equiv x > 0 \wedge true$ so that $\rho^{\#} \in \| x > 0 \wedge true \|^u$. Analogously, $\{(T, \rho^{\#})\} = \| < a?n > true \|^u$. The abstract operator is safe, since the existence of both concrete transitions $(T, \rho_1) \overset{c?n}{\mapsto} (T_1[n/x], \rho_1)$ and $(T, \rho_2) \overset{a?n}{\mapsto} (T_2[n/x], \rho_2)$ for $\rho_1, \rho_2 \in \gamma_e(\rho^{\#})$ is captured. Note that in the abstract process non-determinism among the two actions $c?n$ and $a?n$ arise even if in the concrete case these are two alternative choices.

In contrast, let us consider the lower approximation for formula A. Since there exists $\rho_1, \rho_2 \in \gamma_e(\rho^{\#})$ such that $(T, \rho_1) \overset{a?n}{\not\mapsto}$ and $(T, \rho_2) \overset{c?n}{\not\mapsto}$ the lower approximation is safe if and only if $(T, \rho^{\#}) \notin \| A \|^l$. Since $\| x > 0 \|^l = \emptyset$, $\| x \leq 0 \|^l = \emptyset$ and $c?n = a?x \equiv false$, $\rho^{\#} \notin \| (x \geq 0 \vee true) \wedge (x < 0 \vee false) \|^l$. Therefore, we have both $\| < c?n > true \|^l = \emptyset$ and $\| < a?n > true \|^l = \emptyset$. The abstract operator is able to observe that there is no real non-determinism between $a?n$ and $c?n$, while this is an alternative choice as expressed by $\otimes$ and there are concrete processes for both alternatives.

On the other hand, consider the values abstraction with $\alpha_v(n) = Pos$, if $n > 0$, and $\alpha_v(n) = Neg$, otherwise. There are two abstract environments $\rho_1^{\#}$, with $\rho_1^{\#}(x) = Pos$, and $\rho_2^{\#}$, with $\rho_2^{\#}(x) = Neg$. Since for each $\rho \in \gamma(\rho_1^{\#})$, $\rho \in \| c \|$ for $c = (x > 0 \supset true) \wedge (x \leq 0 \supset false)$, then $\rho_1^{\#} \in \| c \|^l$ is safe. With this evaluation of constraint we can safely obtain $\| < c?n > true \|^l = \{(T, \rho_1^{\#})\}$. Since, for each concrete environment of $\rho_1^{\#}$ only the alternative $x > 0$ is possible, it is safe to conclude that all concrete processes indeed perform $c?n$.

Lemma 6.4. *Let $\| c \|^l$ and $\| c \|^u$ be safe lower and upper approximations for $c \in \mathcal{C}$. Moreover, let $\in_{\equiv}^l$ and $\in_{\equiv}^u$ be safe lower and upper approximations of $\in_{\equiv}$. For each $S^{\#} \in \mathcal{P}(D^{\#})$ and $K \in \{\tau, c?V, c!V\}$, $\alpha^u(\| < K > \|^S\ (\gamma(S^{\#}))) \subseteq \| < K > \|^u\ (S^{\#})$ and $\| < K > \|^l\ (S^{\#}) \subseteq \alpha^l(\| < K > \|^S)(\gamma(S^{\#}))$.*

By lemma 6.4 the lower approximation of the full logic is safe.

Theorem 6.5. *Let* $\| c \|^l$ *and* $\| c \|^u$ *be safe lower and upper approximations for* $c \in \mathcal{C}$. *Moreover, let* $\in^l_{\equiv}$ *and* $\in^u_{\equiv}$ *be safe lower and upper approximations of* $\in_{\equiv}$. *For each closed* μ*-calculus formula* A, $\| A \|^S \supseteq \gamma(\| A \|^l)$.

7 About optimality

We have defined a method for constructing safe dual approximations of the next modality, that exploit dual approximations of constraints and of relation $\in_{\equiv}$. It this setting precision of abstract model checking depends on precision of these approximations. An interesting problem is that of finding conditions on the approximations of constraints and of $\in_{\equiv}$, that guarantee optimality of next modalities. It turns out that optimality of the upper approximation of constraint $FREE_{K,i,j_i}$ and of the lower approximation of constraint $CON_{K,I}$ is sufficient, whenever $\mathcal{SP}(p)$ contains only general processes.

Lemma 7.1. *Let* p *be a closed process such that* $\mathcal{SP}(p) = (GP^*, T^*, \stackrel{\otimes_{i\in\{1,n\}}\Theta_i}{\longrightarrow})$ *where* $GP^* = T^*$. *Let* $\| c \|^u$ *and* $\| c \|^l$ *be optimal upper and lower approximations for* $c \in \mathcal{C}$. *For each* $S^\# \in \mathcal{P}(D^\#)$ *and* $K \in \{\tau, c?V, c!V\}$, $\alpha^u(\| < K > \|^S (\gamma(S^\#))) \supseteq \| < K > \|^u (S^\#)$ *and* $\alpha^l(\| < K > \|^S (\gamma(S^\#))) \subseteq \| < K > \|^l (S^\#)$.

In the upper approximation case this result is quite obvious, since by optimality $\rho^\# \in \| FREE_{a,i,j_i} \|^u$ implies the existence of an environment such that $\rho \in \| FREE_{a,i,j_i} \|$, namely the existence of a corresponding concrete transition. The lower approximation is optimal, whenever there exists a constrained transition if and only if for each concrete process there exists a corresponding concrete transition. Suppose that for each concrete process $(T, \rho) \stackrel{a}{\mapsto} (T', \rho')$. For each ρ there exists a choice such that constraint c_i is satisfied and a non-deterministic choice such that $\rho \models \theta_{i,j_i} = a$. Optimality is guaranteed, since theorem 4.10 ensures in addition that for each ρ all others alternatives c_j are not true. Therefore, for each environment the constraint $CON_{a,I}$ is indeed satisfied so that by optimality the corresponding abstract transition is certainly considered.

In contrast, if $\mathcal{SP}(p)$ contains non-general processes lemma 7.1 is no longer valid. Problems arise from the abstract evaluation of parameters included in the definition of $\in_{\equiv}$. It is sufficient to consider the case of $\rho^\#$, such that $\exists \rho_1, \rho_2 \in \gamma_e(\rho^\#)$ such that $\rho_1 \models FREE_{a,i,j_i}$ and $(T_{i,j_i}, \rho_2) \in_{\equiv} \gamma(S^\#)$, but $\rho_1 \neq \rho_2$.

Depending on the domain of values and expressions specific solutions must be studied for approximating constraints and $\in_{\equiv}$. We suggest a general strategy. Safe dual approximations of constraints can be found on the basis of dual approximations of basic constraints be, $e = e$ and $e \in V$ and by combining lower and upper approximations in the obvious way.

Definition 7.2. For $c \in \mathcal{C}$, let $\| \neg c \|^u = \mathcal{P}(Env^\#) \setminus \| \neg c \|^l$, $\| \neg c \|^l = \mathcal{P}(Env^\#) \setminus \| \neg c \|^u$, $\| c_1 \wedge c_2 \|^l = \cap_{i\in\{1,2\}} \| c_i \|^l$ and $\| c_1 \wedge c_2 \|^u = \cap_{i\in\{1,2\}} \| c_i \|^u$.

If basic constraints are safely approximate these approximations are obviously safe. Unfortunately, they are in general non-optimal, since α^u does not preserve $\cap$, while α^l does not preserve $\cup$.

Moreover, we have to compute dual approximations of $\in_{\equiv}$, that realizes parameters evaluation. Let $gen^{\#}_{T,GP} : Env^{\#} \to \mathcal{P}(Env^{\#})$ be a safe approximation of $gen_{T,GP}$, where $gen_{T,GP}(\rho) \in \gamma_e(gen^{\#}_{T,GP}(\rho^{\#}))$ for each $\rho \in \gamma_e(\rho^{\#})$. This function can be suitably used to define approximations of $\in_{\equiv}$.

Definition 7.3. Let $gen^{\#}_{T,GP}$ be a safe approximation. We define

1. $(T, \rho^{\#}) \in^u_{\equiv} S^{\#}$ iff there exists $\rho^{\#}_1 \in gen^{\#}_{T,GP}(\rho^{\#})$ such that $(GP, \rho^{\#}_1) \in S^{\#}$;
2. $(T, \rho^{\#}) \in^l_{\equiv} S^{\#}$ iff for each $\rho^{\#}_1 \in gen^{\#}_{T,GP}(\rho^{\#})$, $(GP, \rho^{\#}_1) \in S^{\#}$.

If $gen^{\#}_{T,GP}$ is safe, then the previous approximations of $\in_{\equiv}$ are safe.

The difficulties for computing optimal dual approximations are obvious. However, it is important to stress the essential role of $\otimes$ and $\oplus$ for limiting the loss of information the lower approximation. The relations $\otimes$ and $\oplus$ allows to improve the precision of the lower approximation with respect to the "trivial" definition, where a lower approximation of constraint $FREE_{K,i,j_i}$ is considered.

Example 7.4. Consider the process $T_1 \times T_2$ with $T_1 = x > 0 \bigtriangledown c?x.T, a?x.T$ and $T_2 = c!y + 1.T + a!y - 1.T$ of example 4.3. With respect to the abstraction $\alpha_v(n) = \bullet$, for each $n \in Nat$, we trivially obtain $\rho^{\#} \in \| c \|^l$ for $c = (x > 0 \supset (\tau = \tau)) \wedge (x \leq 0 \supset (\tau = \tau))$ and $\rho^{\#}(x) = \bullet$, since $\tau = \tau \equiv true$. Therefore, $(T_1 \times T_2, \rho^{\#}) \in \| < \tau > true \|^l$ is established. The lower approximation of constraint c captures that there are processes for both alternatives, but in both cases the action τ can be performed. In other words, a constrained transition $(T_1 \times T_2, \rho^{\#}) \overset{\tau}{\mapsto} \ldots$ is constructed. If we consider the trivial definition of lower approximation, we would not able to prove it, since neither $\rho^{\#} \in \| (x > 0 \wedge (\tau = \tau)) \|^l$ nor $\rho^{\#} \in \| (x \leq 0 \wedge (\tau = \tau)) \|^l$, even if the evaluation of constraints is optimal. Thus, this trivial method succeeds only if the alternative choice is the same for each concrete processes. In contrast, due to $\otimes$ and $\oplus$ a weaker condition can be considered and a more precise result is achieved. The proposed method could not give the same results on a classical symbolic semantics, where branching represents both non-deterministic and alternative choices, since it exploits the existence of exactly one alternative choice for each process and the representation of all non-deterministic choices for each possibility.

8 Related works

The combination of abstract interpretation and model checking has been the topic of intensive research in the last few years. Much of the work concerned the definition of safe abstract model, namely of safe abstract transition relations, that preserves universal properties [2, 3, 1, 16] and both universal and existential properties [10, 9, 12, 4, 14]. [10, 9, 4] propose the use of constrained and free transitions to handle branching modalities and [9] tackles also the problem of effectively computing for very simple programs an abstract model. The proposed method suffers of the problems of the trivial definition shown in example 7.4. A slight different approach is the one of Kelb [12], that investigates conditions for the safeness of abstract μ-calculus semantics instead that for the safeness of abstract models. In order to handle non-monotonic negation the combination of dual approximations is suggested. However, the problem of computing safe dual approximations of the next modality even over a given abstract model is not addressed. In the framework of value-passing concurrent processes [5] proposes an abstract labelled transition system semantics for abstract closed processes obtained by an abstraction on the values domain. It is not obvious which class of temporal properties is preserved by the abstract model. In [14] it has been introduced the idea of representing the relation of non-determinism and of alternative choice among actions in order to compute more precise safe and constrained transition relations in the framework of closed abstract processes rather than in a symbolic approach with environments. Schmidt [16] 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 a semantics is adequate for the verification of universal properties only. As far as concern the symbolic semantics, it is worth mentioning that other approaches have been proposed for representing regular processes by finite graphs [15].

9 Conclusions

In this paper we have applied abstract interpretation to the verification of μ-calculus properties of value-passing concurrent processes. The main contribution is the definition of a finite symbolic semantics of processes, that differs from the classical one [11] in some aspects. First, classical branching of transitions is replaced by explicit relations of alternative and non-deterministic choices among transitions. Moreover, infinite branches are avoided by representing current recursive calls by means of general processes. The concrete semantics of μ-calculus can suitably be computed by interpreting the symbolic graph over concrete environments, but due to infinite values it is not effectively computable. We have proposed a technique to compute a lower approximate semantics on the symbolic graph by replacing concrete environments with abstract environments. Following the approach of [12] for explicitly treating negation the lower approximate semantics has been obtained by combining lower and upper approximations for each operator of the logic except negation. The relations of non-deterministic and alternative choices turn out to be very useful to approximate the next modality. The lower approximation in particular results undoubtedly more precise than previous proposals [10, 9].

With respect to the classical approach to abstract model checking the proposed method does not rely on the construction of a safe abstract model, but on the computation of safe approximations of the "model checking" functions over the symbolic graph. This approach has several advantages. In order to prove a property it would be typically necessary to subsequently refine the chosen value abstraction by adding more information. This way the construction of the new abstract model is avoided. Moreover, this approach fits well in the traditional abstract interpretation framework and allows us to reason about safety and precision without introducing ad-hoc conditions as for instance the approximation ordering between abstract models of [4, 9]. Recently, Schmidt [17] following the ideas of [18] has pointed out the very close connection among abstract model checking and data-flow analysis. These results suggest that the methods from one area can be usefully used in the other. There are many directions to continue this research. For instance, it seems interesting to study whether the refinement operators [7] that have been designed to systematically construct new more precise domain can be applied to the abstract model checking framework.

Acknowledgments I would like to thank Radhia and Patrick Cousot and Dave Schmidt for their helpful suggestions on 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 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.
6. 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.
7. P. Cousot and R. Cousot. Systematic Design of Program Analysis Frameworks. In *Proc. Sixth ACM Symp. Principles of Programming Languages*, pages 269–282, 1979.
8. D. Dams, R. Gerth, and O. Grumberg. Generation of reduced models for checking fragments of CTL. In *Proceedings of CAV 93*, volume 697 of *Lecture Notes in Computer Science*, pages 479–490. Springer-Verlag, Berlin, 1993.
9. D. Dams, R. Gerth, and O. Grumberg. Abstract Interpretation of Reactive Systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, 1997.
10. 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.

11. M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138:353–389, 1995.
12. P. Kelb. Model Checking and Abstraction: A framework preserving both truth and failure information. Technical report, OFFIS, Oldenburg, Germany, 1994.
13. D. Kozen. Results on the Propositional mu-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.
14. 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.
15. 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.
16. 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.
17. D.A. Schmidt. Data Flow Analysis is Model Checking of Abstract Interpretation. In *Proc. of the Annual ACM Symp. on Principles of Programming Languages*, pages 38–48. ACM Press, 1998.
18. B. Steffen. Data Flow Analysis as Model Checking. In A. Meyer, editor, *Proceedings of Theoretical Aspects of Computer Software (TACS 91)*, volume 526 of *Lecture Notes in Computer Science*, pages 346–364. Springer-Verlag, Berlin, 1991.

A Appendix

Table 1. The concrete semantics

$$\tau.p \overset{\tau}{\mapsto} p \quad c?x.p \overset{c?v}{\mapsto} p[v/x]\ v \in Val \quad c!e.p \overset{c!v}{\mapsto} p\ \mathcal{S}_v(e) = v$$

$$\frac{p_i \overset{a}{\mapsto} p_i'}{p_1 + p_2 \overset{a}{\mapsto} p_i'} + \quad \frac{p_1 \overset{a}{\mapsto} p_1'}{p_1 \times p_2 \overset{a}{\mapsto} p_1' \times p_2}\ int_1 \quad \frac{p_2 \overset{a}{\mapsto} p_2'}{p_1 \times p_2 \overset{a}{\mapsto} p_1 \times p_2'}\ int_2$$

$$\frac{p_1 \overset{a}{\mapsto} p_1'\ p_2 \overset{\bar{a}}{\mapsto} p_2'}{p_1 \times p_2 \overset{\tau}{\mapsto} p_1' \times p_2'}\ sync \quad \frac{T[\bar{e}/\bar{x}] \overset{a}{\mapsto} p}{P(\bar{e}) \overset{a}{\mapsto} p}\ rec\ P(\bar{x}) \equiv T \quad \frac{p_1 \overset{a}{\mapsto} p_1'}{be \bigtriangledown p_1, p_2 \overset{a}{\mapsto} p_1'} \bigtriangledown_1 \mathcal{S}_b(be) = tt$$

$$\frac{p_2 \overset{a}{\mapsto} p_2'}{be \bigtriangledown p_1, p_2 \overset{a}{\mapsto} p_2'} \bigtriangledown_2 \mathcal{S}_b(be) = ff \quad \frac{p \overset{a}{\mapsto} p'}{p \setminus L \overset{a}{\mapsto} p' \setminus L} \setminus chan(a) \cap L = \emptyset$$

The semantic rules for the symbolic semantics are based on some operations over actions. 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 $\Omega_{i,j_i} = T_{i,j_i,1} \oplus \ldots \oplus T_{i,j_i,k_{j_i}}$, for $i \in \{1,2\}$, $j_i \in \{1, n_i\}$, we define

- $\Theta^+_{j_1,j_2} = (c_{1,j_1} \wedge c_{2,j_2}, \oplus_{h_1 \in \{K_1\}} \theta_{1,j_1,h_1} \oplus_{h_2 \in \{1,K_2\}} \theta_{1,j_2,h_2} \oplus \{*\})$ and $\Omega^+_{j_1,j_2} = \oplus_{h_1 \in \{K_1\}} T_{1,j_1} \oplus_{h_2 \in \{1,K_2\}} T_{1,j_2} \oplus T_1 + T_2$, for $K_i = \{h_i \in \{1, k_{j_i}\} \mid \theta_{i,j_i,h_i} \neq *\}$;

- $\bar{\Theta}_{1,j_1} = (be \wedge c_{1,j_1}, \theta_{1,j_1,1} \oplus \ldots \oplus \theta_{1,j_1,k_{j_1}})$ and $\bar{\Theta}_{2,j_2} = (\neg be \wedge c_{2,j_2}, \theta_{2,j_2,1} \oplus \ldots \oplus \theta_{2,j_2,k_{j_2}})$, and $\bar{\Omega}_{i,j_i} = \oplus_{h_i \in \{1,k_{j_i}\}} \bar{T}_{i,j_i,h_i}$ with $\bar{T}_{i,j_i,h_i} = be \bigtriangledown T_1, T_2$ if $\theta_{i,j_i,h_i} = *$ and $\bar{T}_{i,j_i,h_i} = T_{i,j_i,h_i}$ otherwise;
- $\Theta^{\times}_{j_1,j_2} = (c_{1,j_1} \wedge c_{2,j_2}, \oplus_{h_i \in \{1,k_{j_i}\}}\ \theta_{1,j_1,h_1} \times \theta_{2,j_2,h_2})$ and $\Omega^{\times}_{j_1,j_2} = \oplus_{h_i \in \{1,k_{j_i}\}} T'_{1,j_1,h_1} \times T'_{2,j_2,h_2}$ where
  1. if $\theta_{1,j_1,h_1} \in \{\tau, c!e\}$ and $\theta_{2,j_2,h_2} = *$, then $\theta_{1,j_1,h_1} \times \theta_{2,j_2,h_2} = \theta_{1,j_1,h_1}$, $T_{1,j_1,h_1} = T'_{1,j_1,h_1}$ and $T'_{2,j_2,h_2} = T_2$ (**interleaving**);
  2. if $\theta_{1,j_1,h_1} = c?x$ and $\theta_{2,j_2,h_2} = *$ then $\theta_{1,j_1,h_1} \times \theta_{2,j_2,h_2} = c?z$, such that T_{1,j_1,h_1} is free for $[z/x]$ and T_2 is free for z. Moreover, $T'_{1,j_1,h_1} = T_{1,j_1,h_1}[z/x]$ and $T'_{2,j_2,h_2} = T_2$ (**interleaving**);
  3. if $\theta_{1,j_1,h_1} = \theta_{2,j_2,j_2} = *$, then $\theta_{1,h_1,h_1} \times \theta_{2,j_2,h_2} = *$ and $T'_{1,j_1,h_1} = T_1$ and $T'_{2,j_2,h_2} = T_2$ (**idle action**);
  4. $\theta_{1,j_1,h_1} = c!e$ and $\theta_{2,j_2,h_2} = c?x$, then $\theta_{1,j_1,h_1} \times \theta_{2,j_2,h_2} = \tau$ and $T'_{1,j_1,h_1} = T_{1,j_1,h_1}$ and $T'_{2,j_2,h_2} = T_{2,j_2,h_2}[e/x]$ (**synchronization**);
  5. symmetric rules;
- for $\Theta_i = (c_i, \theta_{i,1} \oplus \ldots \oplus \theta_{i,n_i})$ and $\Omega_i = T_{i,1} \oplus \ldots \oplus T_{i,n_i}$, $\Theta^{\setminus}_i = (c_i, \oplus_{j_i \in K_i} \theta_{i,j_i})$ and $K_i = \{j_i \in \{1, n_i\} \mid chan(\theta_{i,j_i}) \cap L = \emptyset$ and $\Omega^{\setminus}_i = \oplus_{j_i \in K_i} T_{i,j_i} \setminus L$.

Table 2. The symbolic semantics

$$nil \xrightarrow{(true,*)} nil \qquad a.T \xrightarrow{(true, a \oplus *)} T \oplus a.T \quad a \in \{\tau, c!e, c?x\}$$

$$\frac{\{T_i \xrightarrow{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{T_1 + T_2 \xrightarrow{\otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \Theta^{+}_{j_1,j_2}} \otimes_{i \in \{1,2\}, j \in \{1,n_i\}} \Omega^{+}_{j_1,j_2}} +$$

$$\frac{\{T_i \xrightarrow{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{be \bigtriangledown T_1, T_2 \xrightarrow{\otimes_{j_1 \in \{1,n_1\}} \bar{\Theta}_{1,j_1} \otimes_{j_2 \in \{1,n_2\}} \bar{\Theta}_{2,j_2}} \otimes_{j_1 \in \{1,n_1\}} \bar{\Omega}_{1,j_1} \otimes_{j_2 \in \{1,n_2\}} \bar{\Omega}_{2,j_2}} \bigtriangledown$$

$$\frac{\{T_i \xrightarrow{\otimes_{j_i \in \{1,n_i\}} \Theta_{i,j_i}} \otimes_{j_i \in \{1,n_i\}} \Omega_{i,j_i}\}_{i \in \{1,2\}}}{T_1 \times T_2 \xrightarrow{\otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \Theta^{\times}_{j_1,j_2}} \otimes_{i \in \{1,2\}, j_i \in \{1,n_i\}} \Omega^{\times}_{j_1,j_2}} \times$$

$$\frac{T[\bar{e}/\bar{x}] \xrightarrow{\otimes_{i \in \{1,n\}} \Theta_i} \otimes_{i \in \{1,n\}} \Omega_i}{P(\bar{e}) \xrightarrow{\otimes_{i \in \{1,n\}} \Theta_i} \otimes_{i \in \{1,n\}} \Omega_i}\ rec\ P(\bar{x}) \equiv T \qquad \frac{T \xrightarrow{\otimes_{i \in \{1,n\}} \Theta_i} \otimes_{i \in \{1,n\}} \Omega_i}{T \setminus L \xrightarrow{\otimes_{i \in \{1,n\}} \Theta^{\setminus}_i} \otimes_{i \in \{1,n\}} \Omega^{\setminus}_i} \setminus$$