Using Acceleration to Compute Parameterized System Refinement

Françoise Bellegarde, Célina Charlet, Olga Kouchnarenko

???0 ???

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Using Acceleration to Compute Parameterized System Refinement

Françoise Bellegarde — Célina Charlet — Olga Kouchnarenko

N° 4716

2003, January

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4716--FR+ENG

INRIA
LORRAINE

Using Acceleration to Compute Parameterized System Refinement

Françoise Bellegarde* , Célina Charlet† , Olga Kouchnarenko‡

Thème 2 — Génie logiciel
et calcul symbolique
Projet CASSIS

Rapport de recherche n° 4716 — 2003, January — 18 pages

Abstract: In this paper, we present a verification approach for a class of parameterized systems. These systems are composed of an arbitrary number of similar processes. As in [4] we represent the states by regular languages and the transitions by transducers over regular languages. If we can compute a symbolic model by acceleration of the actions, then we can also verify a refinement relation $\mathcal{R}$ between the symbolic models. We show that, under some conditions, if $\mathcal{R}$ is verified between two symbolic models, then refinement is verified between concrete parameterized systems. Then, we can take advantage the property (safety and $PLTL$ properties) preservation by refinement for their verification.

Key-words: parameterized systems, refinement, acceleration, verification, safety property

* LIFC – Université de Franche-Comté, email: bellegar@univ-fcomte.fr
† LIFC – Université de Franche-Comté, email: charlet@univ-fcomte.fr
‡ LIFC – Université de Franche-Comté, email: kouchna@univ-fcomte.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Accélérer pour calculer un raffinement de systèmes paramétrés

Résumé : Nous présentons une approche de vérification pour une classe de systèmes paramétrés : ceux composés d'un nombre arbitraire de processus similaires. Pour cela, nous représentons états et transitions de ces systèmes respectivement par des langages réguliers et de transducteurs entre langages réguliers. Si nous pouvons calculer des modèles symboliques de ces systèmes à l'aide d'une technique d'accélération des transducteurs, alors nous pouvons également vérifier une relation de raffinement $\mathcal{R}$ entre ces modèles symboliques. Nous montrons que, sous certaines conditions, si $\mathcal{R}$ est vérifié entre deux modèles symboliques, alors le raffinement est vérifié entre les systèmes paramétrés dont ils sont les modèles. Ceci nous permet de repousser les limites de la vérification des systèmes paramétrés en nous servant de la préservation des propriétés (de sûreté, $PLTL$) par le raffinement.

Mots-clés : systèmes paramétrés, raffinement, accélération, vérification, propriétés de sûreté

1 Introduction

The growing part of computer systems in critical applications points out the importance of developing techniques for verifying these systems. There are two main techniques of verification: *proof* techniques, dealing with infinite state systems but requiring human interaction, and *model-checking* techniques, completely automatic but often restricted to finite state systems [7].

Complex systems verification must take into account aspects that are out of the scope of techniques and tools based on finite state systems analysis (model-checking). Among these aspects, we can distinguish: manipulation of unbounded data structures (e.g counters) or variables [11, 15, 6], parametrization (networks with an arbitrary number of components) [18, 19, 3, 4, 17], mobility, ...

To deal with the problem of infinite systems verification there are several approaches based on varied techniques such as abstraction [14], symbolic reachability analysis, ..., and using varied mathematical tools: languages and automata theory [21, 14, 25, 3, 12], logics, rewriting systems, constraints solving [15, 6], ...

In this paper, we are interested in the verification of parameterized systems. A parameterized system is a particular infinite system in the sense that each of its instances is finite but the number of states of the system depends on one or several parameters. So, taking into account the set of all possible instances of the system requires considering an arbitrary number of cases. Recently, verification of such parameterized systems has been in full expansion [6, 12, 19, 20, 25, 24, 8, 23].

For classes of systems composed of an arbitrary number of processes communicating in a synchronous [18, 19] or asynchronous [17] way and eventually sharing a control process, the problem of verifying a restricted set of properties is decidable. Presently, the most popular techniques for verifying parameterized systems are incomplete [21, 4, 12, 20, 24].

An approach to apply model-checking to a parameterized system is to use a symbolic representation of the states and transitions of the system to generate a finite state abstraction of the operational model of the system.

We suppose a specification obtained by a refinement process. Several methods, like *B* [5], *TLA*$^+$ [22], *CSP2B* [13], ... propose a refinement based development. In such a development, the system conception is done progressively by increasing the accuracy of the systems description at each specification step. The interest of a refinement based development is that it guarantees preservation of the abstract systems properties. In this paper, our refinement concept is close to the one of the *B* event systems. In the *B* refinement method, the refinement is verified by proof. Safety properties of the systems are expressed by an invariant and are verified by proof too. In [10], we express the refinement semantics as a relation between transition systems that can be verified using an algorithmic method. This allows an algorithmic verification of the refinement by model exploration and a verification of the *PLTL* properties by model-checking.

We consider the class of parameterized systems composed of an arbitrary number of similar processes for which the parameter represents the number of processes. In this framework, there is a modelisation where symbolic states can be seen as regular languages [12, 20] and where the reachability problem can be solved using an acceleration method. On such an abstraction, it is possible to:

- verify safety properties of the system by model-checking;
- verify refinement by model exploration;

- verify properties expressed in Propositional Linear Time Logic ($PLTL$) about the behaviors of the system.

In this paper, we extend the algorithmic verification of the refinement relation between finite state systems defined in [10] to parameterized systems. For that, we define the semantics for this refinement relation we call ρ. But ρ cannot be computed directly since we have to use acceleration techniques [12, 20] to compute a finite symbolic representation of a parameterized system. So we define another refinement relation, $\mathcal{R}$, that can be computed using acceleration techniques. The main contribution of this paper is to guarantee that if $\mathcal{R}$ is verified between two symbolic systems, then ρ is also verified for the corresponding concrete systems. We give some conditions required for this to hold. We also provide an algorithmic verification of $\mathcal{R}$ based on fixed points computations. This algorithmic verification is not guaranteed to terminate because it relies on the constructions of the reachability graphs of the systems which are not guaranteed to terminate. The goal of this work is to take advantage of property preservation by the refinement. Indeed, when the refinement is verified, most of the $PLTL$ properties that hold on the symbolic abstract model are preserved for the refined one.

The paper is organized as follows. In Section 2, we give preliminary notions about transducers and regular languages to model parameterized systems and define our notion of symbolic transitions system. In Section 3, we define the semantics of the refinement relation ρ for parameterized systems. As in general ρ can not be computed, we extend it to $\mathcal{R}$, a relation that can be calculated by accelerations techniques. In Section 4, we show that, under some conditions, if the refinement relation $\mathcal{R}$ holds between two symbolic systems, then ρ holds too. In Section 5, we present how $\mathcal{R}$ can be computed in an iterative way. We end by some concluding remarks, a tool description, and directions for future work in Section 5.

2 Symbolic Model

In this section, we define a symbolic model for transition systems based on regular languages and transducers.

2.1 Representing States and Transitions

Let Σ be a finite alphabet. In our approach, states of the system are words of a regular language $E \subseteq \Sigma^*$ and transitions are relations between these words.

Consider the set of all possible states of finite systems for which the last process is in state a and the others are in state w. This set contains the global configurations $a, wa, wwa, ..., w^{n-1}a, ...$ corresponding to systems respectively composed of $1, 2, 3, ..., n, ...$ processes. A state of a system (or global configuration) is a word $a_1 a_2 ... a_n$ over Σ, where n is the number of processes and each a_i is the state of the i-th process.

The transition relation is a relation between words of the same length. A transition labeled by an action α transforms (according to α) a word into another word by changing the letters corresponding to the processes being transformed by α.

Example 1 *Consider an action transforming the most right process in state w into a process in state r. Applying this action to the configuration $wwwa$ (case of a system composed of 4 processes) transforms it into the configuration $wwra$.*

An action α can be written using the triple $\langle ww, (w \rightarrow r), a \rangle$. The second element of the triple describes the transformation made by the action using a word

of pairs over $\Sigma \times \Sigma$. The first and last elements of the triple describe the context in which the action can be applied using words over Σ, i.e., states of the processes which are not concerned with the action.

If, now, the system is composed of an arbitrary number of processes, the regular expression w^*a represents the global state of the system for which the last process is in state a and every other process is in state w, this independently of the number of processes. Then, w^*a characterizes a symbolic *configuration* (global state) of the parameterized system.

We extend the transition relation between words to a symbolic transition relation between regular languages. Consider a symbolic action transforming the first process in state w into a process in state r. Applying this action to a symbolic configuration like w^*a transforms it into the symbolic configuration w^*ra. Such an action can be represented by the triple $\langle \Sigma^*, (w \rightarrow r), \bar{w}^* \rangle$ where $\bar{w} = \Sigma \setminus \{w\}$. The first and last elements of the triple are regular languages, called *left* and *right contexts* of the action. Their definition, according to [4], is as follows:

Definition 1 *A left context C_l is a regular language accepted by a deterministic finite state automaton having a single accepting state. A right context C_r is a regular language such that its reverse language is a left context.*

An action α is represented by a triple $\langle C_l, T, C_r \rangle$ where C_l is a left context, T is a regular language over $\Sigma \times \Sigma$, and C_r is a right context. For example, the symbolic action $\langle w^*, [(r \rightarrow a)(r \rightarrow r)^*(a \rightarrow w)], \epsilon \rangle$ indicates that:

- we first observe an arbitrary number of processes in state w, then,
- the first process being in state r changes his state and becomes a (in other words, it rewrites in a),
- the rightmost process must be in state a and rewrites in w,
- all intermediate processes must be in state r and do not change their state.

2.2 Symbolic Transition Systems

We now define the model using a symbolic transition system. In the formal framework of transition systems, a symbolic configuration is called a symbolic state.

Definition 2 *A symbolic transition system is quadruple $\langle \Sigma, E_0, Act, \rightarrow \rangle$ where Σ is a finite alphabet, E_0 is a regular initial configuration on Σ, Act is a set of (names of) symbolic actions, and $\rightarrow$ is a symbolic transition relation ($\rightarrow \subseteq \Sigma^* \times Act \times \Sigma^*$).*

The labeled transition relation $\rightarrow$ is defined by a set of triples (E, α, E') (written $E \xrightarrow{\alpha} E'$).

The transition relation $\rightarrow$ can be extended to a sequence of transitions in the standard way: E' is reachable from E, written $E \rightarrow^* E'$, if there exists a sequence of symbolic states $E = E^0, E^1, ..., E^n = E'$ such that for each $i \geq 0$ we have $E^i \xrightarrow{\alpha_i} E^{i+1}$. We write $E \nrightarrow$ to express that no action can be applied from E.

Let X be a variable such that its domain is included into Σ^*. Let $Att(E_0) = \{E | E_0 \rightarrow^* E\}$ be the set of reachable states from E_0. Let AP_Σ be a set of atomic propositions where each atomic proposition e is either $X = L$ or $X \subseteq L$, where $L \subseteq \Sigma^*$. Let SP_Σ be a set of propositions over Σ where each proposition is defined by the following grammar: $p_1, p_2 := e | p_1 \vee p_2 | \neg p_1$.

Definition 3 *We define a state proposition $p \in SP_\Sigma$ to be valid for a symbolic state E, written $E \models p$, as follows:*

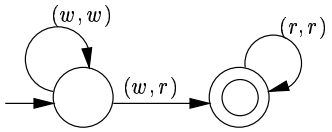

Figure 1: A transducer accepting all words of pairs of the form $(w,w)^*(w,r)(r,r)^*$

- $E \models X = L$ *iff* $E = L$,
- $E \models X \subseteq L$ *iff* $E \subseteq L$,
- $E \models p_1 \vee p_2$ *iff* $E \models p_1$ *or* $E \models p_2$,
- $E \models \neg p$ *iff* $\neg E \models p$.

We call *invariant* a predicate I witch holds on each symbolic state. In other words, I is an invariant of a transition system TS over Σ iff $I \in SP_\Sigma \wedge \forall E.(E \in Att(E_0) \Rightarrow E \models I)$.

2.2.1 Transducers for the transition relation

A transition labeled by α ($\in Act$) can be represented by a finite state automaton over $\Sigma \times \Sigma$, called a transducer (see for example Fig. 1).

Definition 4 *Let Σ_1 and Σ_2 be two alphabets. A transducer $\mathcal{T}$ over $\Sigma_2 \times \Sigma_1$ is a finite state automaton over $\Sigma_2 \times \Sigma_1$ accepting words of pairs (a_i, b_i) for $1 \leq i \leq n$ coming from the words $(a_1 \ldots a_n) \in \Sigma_2^*$ and $(b_1 \ldots b_n) \in \Sigma_1^*$.*

Let $id(L)$ be the language recognized by a transducer accepting identical pairs of words of L. We write id for $id(\Sigma)$. The transducer $\mathcal{T}_\alpha$ for the action $\alpha = \langle C_g, T, C_d \rangle$ accepts all words of pairs belonging to the language defined by $id(C_g).T.id(C_d)$. It expresses the relation $\stackrel{\alpha}{\rightarrow}$.

Definition 5 *The symbolic transition relation $\stackrel{\alpha}{\rightarrow}$ induced by $\mathcal{T}_\alpha$ links all pairs of words $(a_1 ... a_n, b_1 ... b_n)$ such that words of pairs $(a_1, b_1) ... (a_n, b_n)$ are accepted by $\mathcal{T}_\alpha$. We say that $\stackrel{\alpha}{\rightarrow}$ is a regular relation.*

2.3 Calculating the Reachable Configuration for a Transition

Given a transducer $\mathcal{T}_\alpha$ and an automaton $\mathcal{A}$ accepting a symbolic configuration E, we can built a new automaton $\mathcal{A}'$ accepting all words of a symbolic configuration E', such that $E \stackrel{\alpha}{\rightarrow} E'$ and resulting from the composition of $\mathcal{T}_\alpha$ and $\mathcal{A}$, written $\mathcal{T}_\alpha \circ \mathcal{A}$.

Definition 6 *The composition $\mathcal{T}_\alpha \circ \mathcal{A}$ of a transducer $\mathcal{T}_\alpha = \langle \Sigma \times \Sigma, \mathcal{R}, r_0, \delta_\mathcal{T}, F_{\mathcal{T}_\alpha} \rangle$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta_\mathcal{A}, F_\mathcal{A} \rangle$ is the automaton $\langle \Sigma, Q \times \mathcal{R}, (q_0, r_0), \delta, F_A \times F_{\mathcal{T}_\alpha} \rangle$ where $(q_2, r_2) \in \delta((q_1, r_1), b)$ if there exists an $a \in \Sigma$ such that $q_2 \in \delta_\mathcal{A}(q_1, a)$ et $r_2 \in \delta_{\mathcal{T}_\alpha}(r_1, (a, b))$.*

Using this composition, we can compute the next reachable state of the system. But this is not enough to have a finite representation of the behavior of the system because some actions can be executed again and again from an initial configuration. For instance, consider the symbolic state w^*a and the action α transforming the last process in state w into a process in state r. If we apply successively α from w^*a, we get an infinite sequence of symbolic states $w^*a \stackrel{\alpha}{\rightarrow} w^*ra \stackrel{\alpha}{\rightarrow} w^*rra \stackrel{\alpha}{\rightarrow} w^*rrra \stackrel{\alpha}{\rightarrow} ...$

since the transition can be applied again and again from the last computed state. To have a finite representation of the behavior of the system, we must compute in one step the reachable symbolic state corresponding to an arbitrary number of applications of the same action, i.e., compute the transitive closure of the action, called *acceleration*.

2.4 Acceleration

In [4], the authors give the sufficient conditions required to compute automatically the accelerated transducer corresponding to an action transforming only one process, i.e., where T is limited to a pair of letters. In [20, 12], this is extended to actions transforming simultaneously many processes. Let k be the maximal number of times a same symbol can be rewritten for the same action. If k is bounded, an accelerated transducer having a state space in a size of k exponential can be computed. When the construction cannot be calculated, there are widening techniques [12] allowing an upper-approximation of the set of reachable states.

Each accelerated transducer represents the effect of an arbitrary number of applications of the action. We write α^* (resp. α^+) the transitive and reflexive (resp. the transitive closure) of an action α.

The application of an action α (resp. α^+) to a symbolic configuration E is written $\alpha(E)$ (resp. $\alpha^+(E)$). We also write $E(\alpha)$ (resp. $E(\alpha^+)$) the subset of E from which the action α (resp. α^+) can effectively be applied (source of the application). This subset is calculated by intersecting E and the source language of α (resp. α^+). For example, let $E = a^*\{b,c\}a^*$ be a symbolic configuration and $\alpha = \langle \Sigma^*, [b \rightarrow a], \Sigma^* \rangle$ be an action. We have $\alpha(E) = a^* \cup a^*ca^*$ and $E(\alpha) = E \cap \Sigma^* b \Sigma^* = a^*ba^*$.

Using these accelerated actions, we can compute, from the initial symbolic state of the system, the successive sets of reachable symbolic states of the system. If we get a set of configurations that has already been computed (this is detected by an inclusion test between regular languages), then the computation ends. Widening techniques [12] can be used to help termination.

This computation can be viewed as the construction of a reachability graph where the transitions are labeled by α^* or α^+ and where the states are symbolic configurations E_i. This construction is useful to verify the refinement.

3 Refinement

The refinement definition between event systems has been defined in [10] as a relation between transition systems. In this section, we extend this definition to our symbolic models. First, we give a definition of the refinement relation between symbolic transition systems, written ρ. Then, we extend ρ to $\mathcal{R}$ by taking into account transitions acceleration and show that under some conditions, $\mathcal{R}$ and ρ are equal. Finally, we present an iterative computation to verify that $\mathcal{R}$ holds between two given systems.

3.1 Semantics for the Refinement Relation between Symbolic Models

Let $TS_1 = \langle \Sigma_1, E_{01}, Act_1, \rightarrow_1 \rangle$ and $TS_2 = \langle \Sigma_2, E_{02}, Act_2, \rightarrow_2 \rangle$ be two symbolic transition systems at two levels of refinement and I_1, I_2 be their respective invariants. As in [10], we consider that a refinement introduces new actions, so we have $Act_1 \subseteq Act_2$. We also require that $\Sigma_1 \cap \Sigma_2 = \emptyset$, since refinement renames variables.

Our goal is to verify that TS_2 is a refinement of TS_1 by a relation ρ between TS_2 and TS_1. For that, we first define the relation μ, needed to link the states of the systems and from which ρ is a restriction. The relation μ is implemented using a deterministic transducer $\mathcal{T}_\mu$ over $\Sigma_2 \times \Sigma_1$ recognizing the pairs of words $(a_1 \dots a_n, b_1 \dots b_n)$ such that the words of pairs $(a_1, b_1), \dots, (a_n, b_n)$ are accepted by $\mathcal{T}_\mu$. For instance, if $\Sigma_1 = \{a, b\}$ and $\Sigma_2 = \{x, y, z\}$, the configuration $x^*yx^*zx^*$ is linked to the configuration a^*ba^* for the relation $\mu = \{(x, a), (y, b), (z, a)\}$. The transducer $\mathcal{T}_\mu$ accepts the language $((x, a) + (y, b) + (z, a))^*$.

To formalize μ, we also need an invariant $I_{1,2}$ which is a formula of the form $\mathcal{T}(X_2) \subseteq X_1$ where $\mathcal{T}$ is a deterministic transducer over $\Sigma_2 \times \Sigma_1$. The relation μ depends on the validity of the conjunction of I_1, I_2 and $I_{1,2}$.

Definition 7 *We define a proposition p to be valid for the pair of symbolic states (E_2, E_1), written $(E_2, E_1) \models_g p$, as follows:*

- $(E_2, E_1) \models_g p$ *iff either* $p \in SP_{\Sigma_1}$ *and* $E_1 \models p$, *or* $p \in SP_{\Sigma_2}$ *and* $E_2 \models p$ *or,* $p = \mathcal{T}(X_2) \subseteq X_1$ *and* $\mathcal{T}(E_2) \subseteq E_1$,
- $(E_2, E_1) \models_g p \vee q$ *iff* $(E_2, E_1) \models_g p$ *or* $(E_2, E_1) \models_g q$,
- $(E_2, E_1) \models_g \neg p$ *iff* $\neg((E_2, E_1) \models_g p)$.

A symbolic state E_2 in TS_2 is linked to a symbolic state E_1 in TS_1, written $E_2 \ \mu \ E_1$, iff $(E_2, E_1) \models_g I_1 \wedge I_2 \wedge I_{1,2}$. We denote by $\mu(E)$ the language accepted by $\mathcal{T}_\mu(E)$. As in [10], the refinement relation is defined as a restriction of μ.

3.1.1 Definition of the refinement relation.

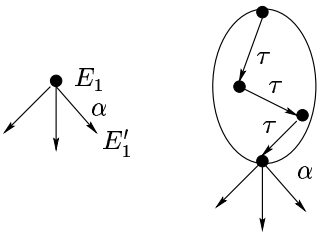

Figure 2: Silent transitions

To describe the refinement relation, we keep the transitions of TS_2 which are labeled in Act_1, but new transitions (those in $Act_2 \setminus Act_1$) that are introduced by the refinement are considered as τ-transitions, i.e. *silent* transitions (see Fig. 2), so Act_2 becomes $Act_1 \cup \{\tau\} = Act_{1_\tau}$. Silent transitions are not allowed to take control of the system forever, so infinite paths are forbidden for silent transitions. We also require that transitions from TS_2 do not introduce new deadlocks. Consequently, vicacity and savety properties are preserved during refinement.

Definition 8 *Let α be an action in Act_1. Let $TS_1 = \langle \Sigma_1, E_{1_0}, Act_1, \rightarrow_1 \rangle$ and $TS_2 = \langle \Sigma_2, E_{2_0}, Act_{1_\tau}, \rightarrow_2 \rangle$ be two transition systems. The refinement relation $\rho \subseteq {\Sigma_2}^* \times {\Sigma_1}^*$ is defined as the greater binary relation included in μ satisfying the following clauses :*

1. **(strict transition refinement)** $(E_2 \ \rho \ E_1 \wedge E_2 \xrightarrow{\alpha}_2 E'_2) \Rightarrow (\exists E'_1 \ s.t. \ E_1 \xrightarrow{\alpha}_1 E'_1 \wedge E'_2 \ \rho \ E'_1)$
   This means that if a symbolic state E_2 of the refined system is linked to a state E_1 of the abstract system and if from E_2 we can apply a transition leading to E'_2, then there must be a state E'_1 in the abstract system such that E'_1 is reachable by applying the transition to E_1, and E'_2 must be linked to E'_1 (see Fig. 3).

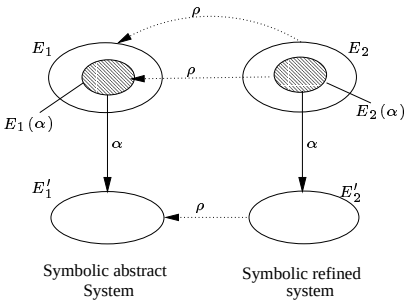

Figure 3: Strict transition

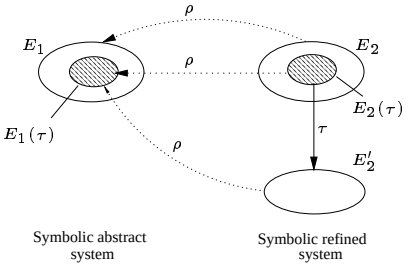

Figure 4: Stuttering transition

2. **(stuttering transition refinement)** $(E_2\ \rho\ E_1 \wedge E_2 \xrightarrow{\tau}_2 E_2') \Rightarrow (E_2'\ \rho\ E_1)$
This means that if a symbolic state E_2 of the refined system is linked to a state E_1 of the abstract system, and if from E_2 we can apply a $\tau-$transition leading to E_2', then E_2' must be linked to E_1 (see Fig. 4).

3. **(non τ-deadlock)** $(E_2\ \rho\ E_1 \wedge E_2 \nrightarrow_2) \Rightarrow (E_1 \nrightarrow_1)$
This means that if a symbolic state E_2 of the refined system is linked to a state E_1 of the abstract system, and if from E_2 no transition can be applied, then no transition can be applied to E_1 either (see Fig. 5).

4. **(non τ-divergence)** $(E_2\ \rho\ E_1) \Rightarrow \neg\ (E_2 \xrightarrow{\tau}_2 E_2' \xrightarrow{\tau}_2 \cdots)$
This means that if a symbolic state E_2 of the refined system is linked to a state E_1 of the abstract system, then we can not have an infinite sequence of τ-transition applying to E_2 (see Fig. 6).

The refinement relation ρ is defined inductively from the initial states of the refined and abstract symbolic transition systems.

Definition 9 *Let $TS_1 = \langle \Sigma_1, E_{10}, Act_1, \rightarrow_1 \rangle$ and $TS_2 = \langle \Sigma_2, E_{20}, Act_{1\tau}, \rightarrow_2 \rangle$ be two transition systems. TS_1 is refined by TS_2, written $TS_1 \sqsupseteq TS_2$, if $E_{20} \rho E_{10}$.*

Since a symbolic state E is a regular language, an action applies to a subset of E. In Fig. 3 and Fig. 4, we see that each configuration represented by $E_2(\alpha)$ is linked to a configuration of the abstract system by ρ. Moreover, $E_1(\alpha)$ equals to $\rho(E_2(\alpha))$.

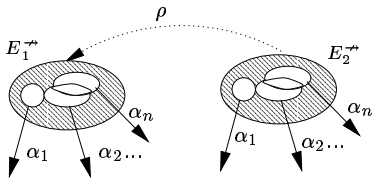

Figure 5: Non τ-deadlock

In Fig. 5, we label with $E_2 \nrightarrow_2$ et $E_1 \nrightarrow_1$ the respective parts of configurations E_2 and E_1 for which no action α_i applies.

Extending ρ to $\mathcal{R}$ by acceleration. Since a symbolic configuration represents an arbitrary number of configurations, a transition may be applied an arbitrary number of times, and the computation of the reachability graph of the system does

not terminate. To help termination, we use acceleration of the actions. We define another relation, $\mathcal{R}$, using strict transition closure. Then, $\mathcal{R}$ is the greater binary relation included in μ and satisfying Clauses 1, 2, 3, 4 of Definition 8 where $\rightarrow$ is replaced by $\stackrel{+}{\rightarrow}$. The computation of $\mathcal{R}$ is more likely to be effective since acceleration helps termination. Using $\mathcal{R}$ instead of ρ to verify refinement is possible iff $E_{20}\rho E_{10}$ is equivalent to $E_{20}\mathcal{R}E_{10}$. Unfortunately, in Section 4, we show that it is not the case. However, we can verify conditions that ensure this equivalence.

4 When Clauses Defining ρ and $\mathcal{R}$ are Equivalent

In order to decide the refinement, we want $E_{20}\rho E_{10}$ and $E_{20}\mathcal{R}E_{10}$ to be equivalent. This is the case if the configurations E_{20} and E_{10} are equivalent for each clause defining ρ and $\mathcal{R}$. In this section, we first show that Clauses 2, 3 and 4 of Definition 8 are equivalent for $\mathcal{R}$ and ρ. Then we show that it is not the case for Clause 1. Finaly we define some conditions, which are easy to verify automatically, under which the equivalence is guaranteed.

4.1 Clauses 2, 3 and 4 of ρ and $\mathcal{R}$ are equivalent

Recall that $\mathcal{R} \subseteq \mu$ and $\rho \subseteq \mu$ and that μ is a relation linking states of transition systems. Then, since relations $\mathcal{R}$ and ρ are defined recursively, we have to put μ in the right part of the implications of the clauses to verify their equivalence. The following three Lemmas expresses the equivalence for Clauses 2, 3 and 4 of relations ρ and $\mathcal{R}$. Proofs for these lemmas can be found in Appendix A.

Lemma 1 *The following propositions 1) and 2) are equivalent:*
1) $(E_2\ \rho\ E_1 \wedge E_2 \stackrel{\tau}{\rightarrow}_2 E'_2) \Rightarrow (E'_2\ \mu\ E_1)$ *2)* $(E_2\ \mathcal{R}\ E_1 \wedge E_2 \stackrel{\tau^+}{\rightarrow}_2 E'^+_2) \Rightarrow (E'^+_2\ \mu\ E_1)$.

This means that if two symbolic states E_2 and E_1 verify the stuttering transition clause for relation ρ (resp. $\mathcal{R}$), then they also verify the stuttering transition clause for relation $\mathcal{R}$ (resp. ρ).

Lemma 2 *The following propositions 1) and 2) are equivalent:*
1) $(E_2\ \rho\ E_1 \wedge E_2 \not\rightarrow_2) \Rightarrow (E_1 \not\rightarrow_1)$ *2)* $(E_2\ \mathcal{R}\ E_1 \wedge E_2 \stackrel{+}{\not\rightarrow}_2) \Rightarrow (E_1 \stackrel{+}{\not\rightarrow}_1)$.

In other words, if two symbolic states E_2 and E_1 verify the clause for no new deadloks for relation ρ (resp. $\mathcal{R}$), then they also verify the clause for no new deadlocks for relation $\mathcal{R}$ (resp. ρ).

Lemma 3 *The following propositions 1) and 2) are equivalent:*
1) $(E_2\ \rho\ E_1) \Rightarrow \neg\ (E_2 \stackrel{\tau}{\rightarrow}_2 E'_2 \stackrel{\tau}{\rightarrow}_2 \cdots)$ *2)* $(E_2\ \mathcal{R}\ E_1) \Rightarrow \neg\ (E_2 \stackrel{\tau^+}{\rightarrow}_2 E'^+_2 \stackrel{\tau^+}{\rightarrow}_2 \cdots)$.

In other words, if two symbolic states E_2 and E_1 verify the non τ-divergence clause for relation ρ (resp. $\mathcal{R}$), then they also verify the non τ-divergence clause for relation $\mathcal{R}$ (resp. ρ).

4.2 Clause 1 of ρ is not equivalent to Clause 1 of $\mathcal{R}$

In this section, we explain why Clause 1 of ρ is not equivalent to Clause 1 of $\mathcal{R}$.

Lemma 4 *Clause 1 of ρ implies Clause 1 of $\mathcal{R}$.*
$((E_2\ \rho\ E_1 \wedge E_2 \xrightarrow{\alpha}_2 E'_2) \Rightarrow (\exists E'_1\ s.t.\ E_1 \xrightarrow{\alpha}_1 E'_1 \wedge E'_2\ \mu\ E'_1)) \Rightarrow ((E_2\ \mathcal{R}\ E_1 \wedge E_2 \xrightarrow{\alpha^+}_2 E'^+_2) \Rightarrow (\exists E'_1\ s.t.\ E_1 \xrightarrow{\alpha^+}_1 E'^+_1 \wedge E'^+_2\ \mu\ E'^+_1))$.

In other words, if two symbolic states E_2 and E_1 verify strict transition refinement clause for relation ρ, then they also verify strict transition refinement clause for relation $\mathcal{R}$.

Lemma 5 *Clause 1 of $\mathcal{R}$ does not imply Clause 1 of ρ.*
$((E_2\ \mathcal{R}\ E_1 \wedge E_2 \xrightarrow{\alpha^+}_2 E'^+_2) \Rightarrow (\exists E'^+_1\ s.t. E_1 \xrightarrow{\alpha^+}_1 E'^+_1 \wedge E'^+_2\ \mu\ E'^+_1)) \not\Rightarrow ((E_2\ \rho\ E_1 \wedge E_2 \xrightarrow{\alpha}_2 E'_2) \Rightarrow (\exists E'_1\ s.t.\ E_1 \xrightarrow{\alpha}_1 E'_1 \wedge E'_2\ \mu\ E'_1))$.

In other words, if two symbolic states E_2 and E_1 verify strict transition refinement clause for relation $\mathcal{R}$, then we can not guarantee that they also verify strict transition refinement clause for relation ρ.

Suppose Clause 1 is satisfied for $\mathcal{R}$. Then, a symbolic state E_2 is linked by μ to a symbolic abstract state E_1. From E_2, we can apply α^+, leading to a E'^+_2. From E_1 we can also apply α^+ and go to state E'^+_1. Moreover, E'^+_2 and E'^+_1 are linked by μ. The refinement relation ρ holds if all intermediate symbolic configurations reachable from E_2 and E_1 by applying successively α are also linked by μ. But, this cannot be guaranteed since we loose the trace of the intermediate symbolic states. So, Clause 1 for ρ is not guaranteed to be satisfied. This is illustrated by Example 2.

Example 2 *Let α_C and α_A be respectively an action of the refined system and an action af the abstract system such that:*

$$\mathcal{L}(\mathcal{T}_{\alpha_C}) = ((a_1,a_1)+(b_1,b_1))^*(b_1,c_1)(b_1,c_1)(c_1,c_1)^*$$
$$\mathcal{L}(\mathcal{T}_{\alpha_A}) = ((a,a)+(b,b))^*(b,c)(c,c)^*$$

Then, using acceleration, we have:

$$\mathcal{L}(\mathcal{T}_{\alpha^+_C}) = ((a_1,a_1)+(b_1,b_1))^*(b_1,c_1)(b_1,c_1)(b_1,c_1)^*(c_1,c_1)^*$$
$$\mathcal{L}(\mathcal{T}_{\alpha^+_A}) = ((a,a)+(b,b))^*(b,c)(b,c)^*(c,c)^*$$

Le μ be the relation such that $\mathcal{L}(\mathcal{T}_\mu) = ((a_1,a)+(b_1,b)+(c_1,c))^*$ and E_{I2}, E_{I1} be two symbolic states linked by μ such that $E_{I2} = (a_1+b_1)^*$ and $E_{I1} = (a+b)^*$.

If we apply α_C to E_{I2}, we get the configuration $\alpha_C(E_{I2}) = (a_1+b_1)^*c_1c_1$. If we apply α_A to E_{I1}, we get the configuration $\alpha_A(E_{I1}) = (a+b)^*c$. These two resulting configurations are not linked by μ because $\mu(\alpha_C(E_{I2})) = (a+b)^*cc$, which is not included in $(a+b)^*c$.

If we apply α^+_C to E_{I2}, we get the configuration $\alpha^+_C(E_{I2}) = (a_1+b_1)^*c_1c_1c_1^*$. If we apply α^+_A to E_{I1}, we get the configuration $\alpha^+_A(E_{I1}) = (a+b)^*cc^*$. These two resulting configurations are linked by μ because $\mu(\alpha^+_C(E_{I2})) = (a+b)^*ccc^*$, which is included in $(a+b)^*cc^*$.

4.3 Ensuring Equivalence

In this section, we show that $E_{20}\rho E_{10}$ is equivalent to $E_{20}\mathcal{R}E_{10}$ under some sufficient conditions. In the preceding section, we saw that Clause 1 verified by $\mathcal{R}$ does not imply that Clause 1 is verified by ρ. But we show that the implication holds if each action of TS_2 *refines* its corresponding action in TS_1.

In our formalism, this means that each pair of words (u,v) recognized by a transducer of the refined system is linked to a pair of words (u',v') recognized by

the corresponding transducer of the abstract system in the following way: $(u, u') \models I_1 \wedge I_2 \wedge I_{1,2}$, and $(v, v') \models I_1 \wedge I_2 \wedge I_{1,2}$, i.e., $\mu(u) = u'$ and $\mu(v) = v'$. To verify that an action of TS_2 *refines* its corresponding action in ST_1, we need the following notations and notions.

Definition 10 *The synchronized product $\mathcal{A}^2$ of an automaton $\mathcal{A} = \langle \Sigma, Q, T, q_0 \rangle$ with itself is the automaton $\langle \Sigma \times \Sigma, Q \times Q, T', (q_0, q_0) \rangle$ where $((q_1, q_2), (e_1, e_2), (q_1', q_2')) \in T'$ iff $(q_1, e_1, q_1') \in T$ and $(q_2, e_2, q_2') \in T$.*

Let $\mathcal{T}$ over $\Sigma_2 \times \Sigma_1$ be a deterministic transducer (a deterministic automaton) corresponding to the relation μ, and $\mathcal{T}^2$ be the synchronized product of $\mathcal{T}$ with itself. $\mathcal{T}^2$ is defined over $((\Sigma_2 \times \Sigma_1) \times (\Sigma_2 \times \Sigma_1))$.

Let f be a function renaming labels of transitions for automata over pairs of pairs of letters like $\mathcal{T}^2$ such that $f : ((\Sigma_2 \times \Sigma_1) \times (\Sigma_2 \times \Sigma_1)) \rightarrow ((\Sigma_2 \times \Sigma_2) \times (\Sigma_1 \times \Sigma_1))$ and $f((e_1, e_2), (e_3, e_4)) = ((e_1, e_3), (e_2, e_4))$. If $\mathcal{T}^2 = \langle ((\Sigma_2 \times \Sigma_2) \times (\Sigma_1 \times \Sigma_1)), Q \times Q, T', (q_0, q_0) \rangle$, then $\mathcal{T}_f^2 = \langle f((\Sigma_2 \times \Sigma_2) \times (\Sigma_1 \times \Sigma_1)), Q \times Q, T', (q_0, q_0) \rangle$. The language accepted by $\mathcal{T}_f^2$ is $\mathcal{L}(\mathcal{T}_f^2) = \{((u, v), (u', v')) | (u, u') \in \mathcal{L}(\mathcal{T}) \wedge (v, v') \in \mathcal{L}(\mathcal{T})\}$.

Consider the composition of the transducer $\mathcal{T}_\alpha$ for an action α of the refined system with our automaton $\mathcal{T}_f^2$ according to Definition 6.

Clause 1 is verified by ρ under the following hypothesis:

Hypothesis 1. *1.* $\forall i.((\alpha_{1i} \in Act_1 \wedge \alpha_{2i} \in Act_2) \Rightarrow \mathcal{L}(\mathcal{T}_{\alpha_{2i}} \circ \mathcal{T}_f^2) \subseteq \mathcal{L}(\mathcal{T}_{\alpha_{1i}}))$.

2. $\forall (u, v).((u, v) \in \mathcal{L}(\mathcal{T}_{\alpha_{2i}}) \Rightarrow \exists (u', v').\ (u, u') \in \mathcal{L}(\mathcal{T}) \wedge (v, v') \in \mathcal{L}(\mathcal{T}))$

This condition says that for all pairs of words (u, v) recognized by a transducer $\mathcal{T}_{\alpha_{2i}}$ corresponding to an action of the refined system, there exists a pair of words (u', v') accepted by the transducer $\mathcal{T}_{\alpha_{1i}}$ corresponding to an action with the same label in the abstract system, such that $(u, u') \in \mathcal{T}$ and $(v, v') \in \mathcal{T}$, i.e, $\mu(u) = u'$ and $\mu(v) = v'$. Example 3 illustrates that Hypothesis 1 is necessary to verify the refinement.

Example 3 *Consider Example 2. We have:*
$\mathcal{L}(\mathcal{T}_{\mu^2_f}(\mathcal{T}_{\alpha_C})) = ((a, a) + (b, b))^*(b, c)(b, c)(c, c)^*$, *which is not included in* $\mathcal{L}(\mathcal{T}_{\alpha_A}) = ((a, a) + (b, b))^*(b, c)(c, c)^*$. *Consequently, Hypothesis 1 does not hold and the refinement is not verified.*

Theorem 1 *Under Hypothesis 1, the strict refinement Clause for ρ is satisfied.*

*Proof idea.*Let α_C be an action refining an action α_A of the abstract system according to Hypothesis 1. Given E_2 and α_C, $E_2' = \{v \mid \exists u \in E_2 \wedge (u, v) \in \alpha_C\}$ and E_2' is not empty. Let $\tilde{u}, \tilde{v}$ be words such that $\tilde{u} \in E_2 \wedge (\tilde{u}, \tilde{v}) \in \alpha_C$. Because of Hypothesis 1, the pair $(\mu(\tilde{u}), \mu(\tilde{v}))$ exists and belongs to α_A. Thus, $\mu(\tilde{u}) \in E_1$, so $\mu(\tilde{v}) \in E_1'$ and E_1' is not empty. By definition, $\mu(E_2') = \{\mu(v) | v \in E_2'\} = \{\mu(v) | \exists u \in E_2 \wedge (u, v) \in \alpha_C\}$. By Hypothesis 1, we also have $\{(u, v) | (u, v) \in \alpha_C\} \subseteq \{(u, v) | (\mu(u), \mu(v)) \in \alpha_A\}$. So, we can write $\mu(E_2') \subseteq \{\mu(v) | \exists u \in E_2 \wedge (\mu(u), \mu(v)) \in \alpha_A\}$. By definition: $\mu(E_2) = \{\mu(u) | u \in E_2\}$, and we can write $\mu(E_2') \subseteq \{\mu(v) | \exists x \in \mu(E_2) \wedge (x, \mu(v)) \in \alpha_A\}$. We also have $\mu(E_2) \subseteq E_1$. Consequently, we can write $\mu(E_2') \subseteq \{\mu(v) | \exists x \in E_1 \wedge (x, \mu(v)) \in \alpha_A\}$. Therefore, $E_1' = \{u | \exists x \in E_1 \wedge (x, u) \in \alpha_A\}$. So, $\mu(E_2') \subseteq E_1'$ and the clause is true.

Theorem 2 *Under Hypothesis 1, $E_{20} \mathcal{R} E_{10}$ is equivalent to $E_{20} \rho E_{10}$.*

Proof. From Theorem 1 and Lemma 4, Clause 1 of ρ is equivalent to Clause 1 of $\mathcal{R}$. Since Lemma 1, 2, 3 guarantee the equivalence between the other clauses, we have

$E_{20}\mathcal{R}E_{10} \Leftrightarrow E_{20}\rho E_{10}$.

Hypothesis 1 can be verified according to Definitions 6 and 10 before the algorithmic verification of the refinement relation $\mathcal{R}$, which is described in the next section.

4.4 Iterative Computation of a τ-simulation

In [10], we show that the refinement relation is a special case of τ-simulation. It is well-known that a (bi)simulation can be computed in an iterative way when one of the systems is finitely branching. In our framework, the transition systems are finitely branching since the sets Act and Σ are finite sets.

Consequently, the relation $\mathcal{R}$ can be calculated by a width-first exploration of the set of reachable symbolic configurations. This exploration uses successive under-approximations of the least fixed points calculation for the set of reachable configurations. To give a semi-effective procedure to decide refinement, we must be able to compute completely the simultaneous exploration of the two systems symbolic reachable configurations. We do this exploration from the refined systems reachable states, since $\mathcal{R}$ is a τ-simulation of the refined system TS_2 by the abstract system TS_1. For that, it is enought to know:

1. How to glue symbolic configurations? We must decide whether the current configuration E_2 of TS_2 is linked by μ to the current configuration or not. E_1 of TS_1. This is effective since we can decide whether $(E_2, E_1) \models_g I_1 \wedge I_2 \wedge I_{1,2}$ or not.

2. How to compute the actions accelerations for the abstract and refined systems to calculate the simulation? This is semi-effective because the computation of the accelerations may not terminate.

3. How to decide strict transition refinement? This if effective by composing the transducers for the actions of the refined system with $\mathcal{T}_f^2$ and checking that either the language of the resulting automaton is included in the language of the transducer of the corresponding action of the abstract system or not.

4. How to decide stuttering transition refinement? For that,

   (a) we have to compute the τ-transitions, i.e, the new transitions accelerations; the calculation of a τ-transition is semi-effective;

   (b) we must construct the symbolic reachable configurations using the τ-transitions from a configuration E_2 already linked to E_1. Remark that this needs a width first exploration of the set of symbolic configurations reachable by the τ-transitions and, consequently, successive upper-approximations of a least fixed point computation.

5. How to decide lack of τ-deadlocks? For that, we detect deadlocks from each symbolic reachable configuration of the refined model during the joint exploration (see Fig 5).

6. How to decide non τ-divergence? A sufficient condition for that is the absence of cycles of τ-transitions in the reachability graph. Detection of such cycles, called τ-cycles, is done using a semi-effective procedure, detailed below.

Within the τ-transitions we have a *potential cycle* when a symbolic configuration E' has a non empty intersection with a symbolic configuration E being one of its predecessors in a sequence of τ-transitions. Let $E = E_0, E_1, \ldots, E_n = E'$ be

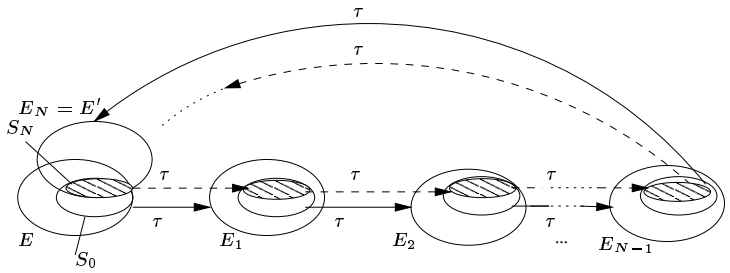

Figure 6: Non τ-divergence

the symbolic configurations of the sequence of τ-transitions from E to E'. Let S_0, S_1, ... , S_n be the source parts of each τ-application of the potential cycle (see Fig. 6). We have the three following cases:

- Case 1. A potential cycle corresponds to a τ-cycle (then it is a fixed point of the sequence of τ-transitions) if $S_n = S_0$ (see Example 4).
- Case 2. A potential cycle does not correspond to any τ-cycle if $S_n \cap S_0 = \emptyset$ (see Example 5) .
- Case 3. Otherwise, we cannot conclude and we must apply the sequence of τ-transitions again from the configuration $S_n \cap S_0$ (see Example 6) .

Example 4 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two transducers for τ-transitions and E_0 be an initial state. We have:*

$$\begin{aligned}\mathcal{L}(\mathcal{T}_1) &= (a \to b)id(\Sigma^*)\\ \mathcal{L}(\mathcal{T}_2) &= (b \to a)id(\Sigma^*)\\ E_0 &= aa^*.\end{aligned}$$

Then, by applying successively $\mathcal{T}_1$ and $\mathcal{T}_2$ from E_0, we get the following sequence: $E_0 = aa^*, E_1 = ba^*, E_2 = aa^*$. Here, the source parts of the τ-transitions are equal to the computed configurations. So, we have $E_0 = S_0 = aa^*$, $E_1 = S_1 = ba^*$, andt $E_2 = S_2 = aa^*$. We have a potential cycle because $E_0 \cap E_2$ is not empty. Moreover, we have $S_0 = S_2$. Consequently the potential cycle corresponds to a τ-cycle.

Example 5 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two transducers for τ-transitions and E_0 be an initial state. We have:*

$$\begin{aligned}\mathcal{L}(\mathcal{T}_1) &= id(w+r)^*(a \to w)id(w+r)^*(r \to a)id(w+r)^*\\ \mathcal{L}(\mathcal{T}_2) &= id(w^*a)(r \to w)((r \to w + id(w))^*\\ E_0 &= (w+r)^*a(w+r)^*.\end{aligned}$$

Then, by applying successively $\mathcal{T}_1$ and $\mathcal{T}_2$ from E_0, we get the following sequence: $E_0 = (w+r)^*a(w+r)^*, E_1 = (w+r)^*w(w+r)^*a(w+r)^*, E_2 = ww^*aw^*$. There is a potential cycle because $E_0 \cap E_2$ is not empty. $E_0 \cap E_2 = ww^*aw^*$. We want to know if this potential cycle is a τ-cycle. For that we compute the source parts of the transducers for the τ-transitions. We get $S_0 = (w+r)^*a(w+r)^*r(w+r)^*$, $S_1 = w^*w(r+w)^*$ and $S_2 = \emptyset$. So the potential cycle is not a τ-cycle.

Example 6 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two transducers for τ-transitions and E_0 be an initial state. We have:*

$$\begin{aligned}\mathcal{L}(\mathcal{T}_1) &= id(a^*a)(b \to c)id(b)^*\\ \mathcal{L}(\mathcal{T}_2) &= id(\Sigma)^*(c \to a)id(b)^*\\ E_0 &= a^*ab^*.\end{aligned}$$

Then, by applying successively $\mathcal{T}_1$ and $\mathcal{T}_2$ from E_0, we get the following sequence: $E_0 = a^*ab^*, E_1 = a^*acb^*, E_2 = a^*aab^*$. As $E_0 \cap E_2$ is not empty, we have a potential cycle. We compute the source parts of the transducers, and we get $S_0 = a^*abb^*$, $S_1 = E_1 = a^*acb^*$, et $S_2 = a^*aabb^*$. As S_2 is not empty, we can not conclude. We have to aply the transducers again from $S_0 \cap S_2 = a^*aabb^*$. By applying the tranducers again, we generate a growing sequence of configurations like $S_2 = a^*aaabb^*$, then $S_2 = a^*aaaabb^*$, etc... and we can not conclude.

The refinement relation provides a formal framework for verifying parameterized systems, since when the refinement holds, most of the properties that have been verified on the abstract model are preserved by the refined one (See [9] for more details).

5 Conclusions and Future Works

In this paper, we show that the refinement relation that we have defined using acceleration can be semi-computed. This allows us to verify the refinement relation for an intersecting class of parameterized systems, those for which the parameter is an arbitrary number of similar processes. The refinement relation is computed on the symbolic models of the systems. These symbolic models must also be computed. For that, we must represent states and actions of the system by regular languages and transducers and compute the accelerations of the actions (it is possible when the actions satisfy the conditions given in [3, 4]). We must also generate a reachability graph where the states regular languages and transitions are accelerated actions by using an effective construction of the set of reachable states. In this paper, we also provide semi-algorithms to 1)calculate the reachability graph of the system 2)detect absence of new τ-cycles. Remark that in our approach, the reachability graph is used by other algorithm or semi-algorithms to verify the refinement clauses.

Tools. We have implemented a tool, using *LASH* [1] libraries to verify the refinement relation for parameterized systems and the tool *RMC* [2] to compute transducers acceleration. Presently, our tool generates symbolic reachability graphs, verifies refinement between two symbolic systems and also verifies safety properties for these systems. When the refinement is not verified, our tool finds the origin of the error. We have used our tool to verify automatically the refinement between two levels of refinement of the PID system [16]. It took 13 seconds (see [9] for more details). We must however complete this framework by adding the verification of *PLTL* properties. When the tool will be completed, we will be in measure to evaluate the contribution of our verification approach for parameterized systems.

References

[1] The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[2] Regular Model Checking (RMC). Available at `http://www.regularmodelchecking.com/`.

[3] P. Abdulla and B. Jonsson. On the existence of network invariants for verifying parametrized systems. In Olderog and Steffen, editors, *Correct System Design – Recent Insights and Advances*, volume 1710 of *LNCS*, pages 180–197. Springer-Verlag, 1999.

[4] P. A. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling global conditions in parametrized system verification. In *Proc. 11th Int. Conf. on*

Computer Aided Verification CAV'99, volume 1633 of LNCS, pages 134–145, 1999.

[5] J.-R. Abrial. *The B Book*. Cambridge University Press - ISBN 0521-496195, 1996.

[6] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning. In *Proc. Int. Conf. on Computer Aided Verification, CAV'00, volume 1855 of Lecture Notes in Computer Science*, pages 419–434. Springer-Verlag, 2000.

[7] K. R. Apt and D. C. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, May 1986.

[8] T. Arons, A. Pnueli, S. Ruah, Y. Xu, and L. Zuck. Parameterized verification with automatically computed inductive assertions. In G. Berry, H.Comon, and A. Finkel, editors, *Proc. Int. Conf. CAV'01, volume 2102 of Lecture Notes in Computer Science*, pages 221–234. Springer Verlag, 2001.

[9] F. Bellegarde, C. Charlet, and O. Kouchnarenko. Raffiner pour vérifier des systèmes paramétrés. *Technique et Science Informatiques*, 8(21):1121–1149, 2002.

[10] F. Bellegarde, J. Julliand, and O. Kouchnarenko. Ready-simulation is not ready to express a modular refinement relation. In *Proc. Int. Conf. on Fondamental Aspects of Software Engineering, FASE'2000*, volume 1783 of *LNCS*, pages 266–283. Springer-Verlag, April 2000.

[11] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Lecture Notes in Computer Science*, volume 818, pages 55–67. Springer-Verlag, 1994.

[12] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In E. A. Emerson and A. P. Sistla, editors, *Proc. Int. Conf. on Computer Aided Verification, CAV'00, volume 1855 of Lecture Notes in Computer Science*, pages 403–418. Springer-Verlag, 2000.

[13] M. J. Butler. csp2B: A practical approach to combining CSP and B. *Formal Aspects of Computing*, 12:182–198, 2000.

[14] E. M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks using abstraction and regular languages. In *Proc. of the 6th International Conference on Concurrency Theory (CONCUR'95), Philadelphia, PA*, pages 395–407, 1995.

[15] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *Proc. Int. Conf. CAV'98, volume 1427 of LNCS*, Springer-Verlag, 1998.

[16] J. Dick and A. Faivre. Automating the generation and sequencing of test cases from model-based specifications. *FME'93: Industrial-Strength Formal Methods*, LNCS 670 Springer-Verlag:268–284, April 1993.

[17] E. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *In 17th International Conference on Automated Deduction (CADE-17)*, volume 14, pages 236–255, 2000.

[18] E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In *Proc. 22nd ACM SIGPLAN-SIGACT Int. Symp. on Principles of Programming Languages (POPL'95)*, pages 85–94, San Francisco, California, January 1995. ACM Press.

[19] E. A. Emerson and K. S. Namjoshi. Automatic verification of parameterized synchronous systems. In *Proc. Int. Conf. CAV'98*, volume 1102 of *LNCS*, 1996.

[20] B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *Proc. 6th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000)*, LNCS, pages 220–234. Springer-Verlag, 2000.

[21] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *In O. Grumberg, editor, Proceedings of the 9th International Conference on Computer Aided Verification (CAV 1997)*, LNCS, pages 424–435. Springer Verlag, 1997.

[22] Leslie Lamport. Specifying concurrent systems with tla +. In *Calculational System Design*, pages 183–247, Amsterdam, 1999. In Manfred Broy and Ralf Steinbruggen editors.

[23] M. Maidl. A unifying model checking approach for safety properties of parameterized systems. In G. Berry, H.Comon, and A. Finkel, editors, *Proc. Int. Conf. CAV'01, volume 2102 of Lecture Notes in Computer Science*, pages 311–323. Springer Verlag, 2001.

[24] A. Pnueli and E. Shahar. Liveness and acceleration in parametrized verification. In *Computer Aided Verification, CAV'00*, volume 1855 of *LNCS*, pages 328–343, 2000.

[25] A. P. Sistla and V. Gyuris. Parameterized verification of linear networks using automata as invariants. *Formal Aspects of Computing*, 11(4):402–425, 1999.

A Equivalence for Clauses Defining Relations $\mathcal{R}$ and ρ.

Lemma [4] *Clause 1 for ρ implies Clause 1 for $\mathcal{R}$.*
$((E_2\ \rho\ E_1 \wedge E_2 \xrightarrow{\alpha}_2 E_2') \Rightarrow (\exists E_1'\ s.t.\ E_1 \xrightarrow{\alpha}_1 E_1' \wedge E_2'\ \mu\ E_1')) \Rightarrow ((E_2\ \mathcal{R}\ E_1 \wedge E_2 \xrightarrow{\alpha^+}_2 E'^+_2) \Rightarrow (\exists E_1'\ s.t. E_1 \xrightarrow{\alpha^+}_1 E'^+_1 \wedge E'^+_2\ \mu\ E'^+_1))$

Proof. We have to proove that (1) $\Rightarrow$ (2) where (1) is $((E_2\ \rho\ E_1 \wedge E_2 \xrightarrow{\alpha}_2 E_2') \Rightarrow (\exists E_1'$ s.t. $E_1 \xrightarrow{\alpha}_1 E_1' \wedge E_2'\ \mu\ E_1'))$ and (2) is $((E_2\ \mathcal{R}\ E_1 \wedge E_2 \xrightarrow{\alpha^+}_2 E'^+_2) \Rightarrow (\exists E_1'$ s.t. $E_1 \xrightarrow{\alpha^+}_1 E'^+_1 \wedge E'^+_2\ \mu\ E'^+_1))$. For that we proove $\neg(2) \Rightarrow \neg(1)$. Suppose $\neg(2)$. There are 2 possibilities:

- First, $\exists E_1'$ s.t. $E_1 \xrightarrow{\alpha^+} E'^+_1$ but $\neg(E'^+_2\,\mu\,E'^+_1)$. By definition $E'^+_2 = \bigcup_{i\geq 1} E_2^i$ s.t. $E_2^{i-1} \xrightarrow{\alpha} E_2^i$ where $E_2^0 = E_2$. So, there exists at least one configuration $E_2^j \subseteq E'^+_2$ s.t. $\neg(E_2^j \mathcal{R} E'^+_1)$. By definition, $E'^+_1 = \bigcup_{i\geq 1} E_1^i$ s.t. $E_1^{i-1} \xrightarrow{\alpha} E_1^i$ where $E_1^0 = E_1$. So, $\neg(E_2^j \mu E_1^j)$ where $E_1^j \subseteq E'^+_1$. Therefore, ρ does not hold.
- Second, there is no E_1' s.t. $E_1 \xrightarrow{\alpha^+}_1 E'^+_1$). So there is no α transition from E_1.

For the other clauses, we show equivalence of the Clauses (1)(for ρ) and (2)(for $\mathcal{R}$) by 2 contrapositions. Case 1 corresponds to prooving that $\neg(1) \Rightarrow \neg(2)$ and case 2 corresponds to prooving that $\neg(2) \Rightarrow \neg(1)$. Recall that $\rho \subseteq \mu$ and $\mathcal{R} \subseteq \mu$.

Lemma [1] *The following propositions are equivalent.*
1) $(E_2\ \rho\ E_1 \wedge E_2 \xrightarrow{\tau}_2 E_2') \Rightarrow (E_2'\ \mu\ E_1)$ *2)* $(E_2\ \mathcal{R}\ E_1 \wedge E_2 \xrightarrow{\tau^+}_2 E'^+_2) \Rightarrow (E'^+_2\ \mu\ E_1)$

Proof. **Case 1.** Suppose $E_2\ \rho\, E_1$, $E_2 \xrightarrow{\tau}_2 E_2'$ and $\neg(E_2'\mu\, E_1)$. By definition $E'^{+}_2 = \bigcup_{i\geq 1} E_2^i$ s.t $E_2^{i-1} \xrightarrow{\tau} E_2^i$ where $E_2^0 = E_2$. So, $E_2' \subseteq E'^{+}_2$. Consequently $\neg(E'^{+}_2 \mu E_1)$ and we are done.
Case 2. Suppose $E_2 \mathcal{R} E_1$, $E_2 \xrightarrow{\tau^+}_2 E'^{+}_2$ and $\neg(E'^{+}_2\, \mu\, E_1)$. By definition $E'^{+}_2 = \bigcup_{i\geq 1} E_2^i$ s.t. $E_2^{i-1} \xrightarrow{\tau} E_2^i$ where $E_2^0 = E_2$. So, there exists E_2^j part of E'^{+}_2 s.t. $\neg(E_2^j \mu E_1)$ and we are done.

Lemma [2] *The following propositions are equivalent.*
1) $(E_2\ \rho\ E_1 \wedge E_2 \nrightarrow_2) \Rightarrow (E_1 \nrightarrow_1)$ *2)* $(E_2\ \mathcal{R}\ E_1 \wedge E_2 \overset{+}{\nrightarrow}_2) \Rightarrow (E_1 \overset{+}{\nrightarrow}_1)$

Proof. **Case 1.** Suppose $E_2\ \rho\ E_1$, $E_2 \nrightarrow_2$ and $\neg(E_1 \nrightarrow_1)$. If no action α can be applied from E_2, then no action α^+ can be applied from E_2. If α can be applied from E_1, then α^+ can also be applied from E_1.
Case 2. Suppose $E_2\ \mathcal{R} E_1$, $E_2 \overset{+}{\nrightarrow}_2$ and $\neg(E_1 \overset{+}{\nrightarrow}_1)$. If no action α^+ can be applied from E_2, then no action α can be applied from E_2. If α^+ can be applied from E_1, then α can also be applied from E_1.

Lemma [3] *The following propositions are equivalent.*
1) $(E_2\ \rho\ E_1) \Rightarrow \neg\ (E_2 \xrightarrow{\tau}_2 E_2' \xrightarrow{\tau}_2 \cdots)$ *2)* $(E_2\ \mathcal{R}\ E_1) \Rightarrow \neg\ (E_2 \xrightarrow{\tau^+}_2 E_2' \xrightarrow{\tau^+}_2 \cdots)$

Proof. **Case 1.** Suppose $E_2 \rho E_1$. Let $\sigma = \tau_1, \tau_2, ..., \tau_n$ be a sequence of τ-transitions such that $E_2^0 \xrightarrow{\tau_1} E_2^1 \xrightarrow{\tau_2} ... \xrightarrow{\tau_n} E_2^n = E_2^i$ with $(i < n)$ inducing a τ-cycle. Let E'^{1}_2 be such that $E_2^0 \xrightarrow{\tau_1^+} E'^{1}_2$. By definition $E'^{1}_2 = \bigcup_{i\geq 1} E_2^i$ such that $E_2^{i-1} \xrightarrow{\tau_1} E_2^i$ so $E_2^1 \subseteq E'^{1}_2$. So, τ_2 (and τ_2^+) can be applied from E'^{1}_2. In the same way, the other transitions can also be applied and the sequence σ' of transitions $\tau_1^+, \tau_2^+, ..., \tau_n^+$ can also be applied. Moreover, $E_2^n \subseteq E'^{n}_2$. So, the sequence σ' induces a cycle and $\mathcal{R}$ is not verified.
Case 2. Suppose $(E_2 \mathcal{R} E_1)$ and $\sigma' = \tau_1^+, \tau_2^+, ..., \tau_n^+$ be a sequence of τ-transitions such that $E'^{0}_2 \xrightarrow{\tau_1^+} E'^{1}_2 \xrightarrow{\tau_2^+} ... \xrightarrow{\tau_n^+} E'^{n}_2 = E'^{i}_2$ with $(i < n)$ inducing a cycle. Obviously, σ' induces a cycle from E'^{0}_2 and we are done.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399