Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 98 (2004) 21–33
www.elsevier.com/locate/entcs

A Symbolic Procedure for Control Reachability in the Asynchronous π-calculus

Extended Abstract

Giorgio Delzanno[1]

Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova, Via Dodecaneso 35 - 16146 Genova, Italy

Abstract

We study the relationship between the asynchronous π-calculus and the specification language MSR$_{NC}$ combining *multiset rewriting over first-order atomic formulas* (MSR) and *name constraints* (NC) proposed in [10]. We exploit this connection to define a sound and fully automatic procedure for attacking *control reachability* for infinite-state specifications given in asynchronous π-calculus, i.e., for specifications of *mobile processes* with unbounded control, name generation, and name mobility.

Keywords: Mobile concurrent systems, Control reachability, Constraints, Symbolic state exploration.

1 Introduction

In [13] German and Sistla established a connection between Petri Nets and CCS by means of which automated verification methods like the *covering graph construction* could be transferred to CCS-like models (see e.g. [5]). In this setting individual processes are viewed as "communicating finite state machines" (with one place buffer), whereas the entire system is composed of an arbitrary (but finite) number of processes. The connection between CCS and Petri Nets has been extended in several different ways. For instance, in

[1] Email: giorgio@disi.unige.it

1571-0661
doi:10.1016/j.entcs.2003.10.004

[12] communication mechanisms like *broadcast* have been modelled via *transfer arcs*. Control reachability is still decidable for the extended Petri Nets models proposed in [12]. Formalisms used to specify *mobile processes*, often called *nominal calculi* [14], represent another important extension of *value passing CCS*. In this setting the use of channel names as values provides for a dynamic reconfiguration of the network (i.e. of the communication links between processes). A well-known example of nominal calculus is the π-calculus [16]. In the π-calculus *process mobility* is achieved by using names as communication ports. Automated verification of specifications in the π-calculus becomes particularly challenging due to the presence of *fresh name generation*, *name mobility*, and *unbounded control*, i.e., their state-space is infinite in *several dimensions*. The application of automatic verification techniques developed for Petri Nets to specifications given in the π-calculus has been explored in different works in the literature. For instance, in [4] control reachability has been shown to be decidable for different *fragments* of asynchronous π-calculus (π_a) via a reduction to Petri Nets with *transfer arcs*. The use of Petri Nets indicates a restriction to models with one infinite dimension (e.g. the number of processes or the number of names). Similar restrictions are taken in other verification methods for mobile systems like, e.g., [17,18,19], where processes are required to be *finitary* (there is a bound on the number of parallel components generated during execution).

The research direction that we are currently investigating concerns the applicability of *infinite-state verification methods* developed for concurrent systems with several sources of infiniteness [10] to mobile processes. Specifically, in this paper we will investigate the connection between specifications of *mobile processes* given in the asynchronous π-calculus and MSR_{NC} [10,11], a specification language based on *multiset rewriting over first order atomic formulas* (MSR) and *name constraints* (NC). MSR_{NC} is a conservative extension of Petri Nets in which tokens are represented via *atomic formulas* and constraints define the relationship over data attached to the tokens. In order to establish a formal connection, we embed the formulation of asynchronous π-calculus proposed in [4] based on the notion of *normalised equations* into MSR_{NC}. The proposed encoding preserves (control) reachability. Furthermore, it allows us to transfer the verification procedures for attacking control reachability studied in the context of MSR_{NC} [10] to mobile processes. Specifically, the verification method is based on a symbolic representation of upward closed sets of configurations of unbound π_a specifications. This data structure can be used then to attack the control reachability problem using *symbolic backward reachability*. In fact, the computation of the *pre-image* of a π_a specification can be made effective by using the encoding and by specializing the pre-image operator defined for MSR_{NC} specifications.

The resulting method gives us a *fully automatic* and *sound* procedure for the verification of safety properties (often reducible to control reachability) for mobile processes. Termination cannot be guaranteed for generic π_a specification. However, techniques like abstract interpretation or heuristics inspired to the Structural Theory of Petri Nets can be used here to enforce termination, to accelerate the speed of the analysis, or to simply compute approximated results. Furthermore, the study of fragments of π_a related to the monadic fragment of MSR_{NC} (for which backward reachability terminates) could represent a promising research line for finding new decidability results for mobile processes.

1.1 *Asynchronous π-calculus (π_a)*

The *asynchronous π-calculus* (π_a) is a subcalculus of the π-calculus without choice and match and in which message emission is non-blocking [15,6]. The set of π_a processes is defined as follows

$$P \quad ::= \quad \mathbf{0} \;\mid\; \overline{x}y \;\mid\; x(y).P \;\mid\; P_1|P_2 \;\mid\; (\nu x)P \;\mid\; !P$$

The term $\mathbf{0}$ denotes a null process. The output term $\overline{x}y$ denotes an asynchronous message with target x and content y. With the input prefix $x(y).P$ a process receives an arbitrary name z at channel x and then behaves like $P[z \mapsto y]$. The process $P[z \mapsto y]$ is the result of substituting all free occurrences of y in P by z. The argument y of $x(y)$ binds all free occurrences of y in P. The composition $P|Q$ consists of P and Q running in parallel. The restriction $(\nu x)P$ behaves like P except that it cannot exchange messages targeted to x with the environment; the argument x of (νx) binds all free occurrences of x in P. The replication $!P$ provides an arbitrary number of copies of process P ($!P \;\equiv\; P \mid !P$).

In [4], Amadio and Meyssonier proposed an equivalent reformulation based on the notion of *normalised* parametric equations in which repetition is replaced by recursion. In this paper we will take it as reference model. Let us use $\vec{a}$ to denote a tuple $a_1, \ldots, a_n$ of names, and $[\vec{a} \mapsto \vec{b}]$ to indicate a substitution mapping a_i to b_i for $i : 1, \ldots, n$. Furthermore, let the term $(\nu\vec{v})P$ denote the term $(\nu v_1)\ldots(\nu v_n)P$. Following [4], a *normalised* parametric equation is defined as follows

$$A(\vec{x}) = \underbrace{a(\vec{u})}_{input} . \underbrace{(\nu\vec{v})}_{gen.} (\underbrace{\overline{a}_1\vec{y}_1 \mid \ \ldots \ \mid \overline{a}_n\vec{y}_n}_{output} \ \mid \ \underbrace{A_1(\vec{w}_1) \ \ldots \ \mid A_m(\vec{w}_m)}_{continuations})$$

where $A, A_1, \ldots$ denote process identifier; the (bound) names in $\vec{x}$, $\vec{u}$, $\vec{v}$ are all distinct each other; $a_i, \vec{y}_i$ and $\vec{w}_j$ are names taken from $\vec{x}, \vec{u}, \vec{v}$ for $i : 1, \ldots, n$ and $j : 1, \ldots, m$. We will use $Fn(P)$ to denote the set of free names in the body of an equation $A(\vec{x}) = P$. A *process* is defined via a set $\mathcal{E}$ of normalised parametric equations, and by an *initial configuration*. A *configuration* is formally defined as a normalised process of the shape

$$(\nu\vec{v})(\underbrace{\overline{a}_1\vec{y}_1 \mid \ldots \mid \overline{a}_n\vec{y}_n}_{messages} \mid \underbrace{A_1(\vec{w}_1) \ \ldots \mid A_m(\vec{w}_m)}_{processes})$$

Two configurations P and Q are equivalent, written $P \equiv Q$, if P is syntactically equal to Q up to renaming of bound names, and associativity and commutativity of parallel composition.

The operational semantics of a process is defined as the reflexive-transitive closure of the reduction relation $\cdot \Rightarrow_{\pi_a} \cdot$ defined over configurations as follows. Let P be the configuration $(\nu\vec{w})(A(\vec{u}) \mid \overline{c}(\vec{v}) \mid Q)$ where Q is a multiset of messages and continuations, and let $D \in \mathcal{E}$ be the equation $A(\vec{x}) = a(\vec{y}).(\nu\vec{z})R$ such that the set of names $\vec{x}$, $\vec{y}$, $\vec{z}$ and $\vec{w}$ and $Fn(P)$ are all distinct each other. Given $\sigma = [\vec{x} \mapsto \vec{u}, \vec{y} \mapsto \vec{v}]$ and its natural extension $\hat{\sigma}$ to expressions, if $\hat{\sigma}(a) = c$, then P reduces to P', written $P \Rightarrow_{\pi_a} P'$, where $P' = (\nu\vec{w}, \vec{z})\ (\hat{\sigma}(R) \mid Q)$.

The *control reachability problem* [4] is defined as follows. Given a set of equations $\mathcal{E}$ containing the process identifier A, and an initial configuration P, does $P \Rightarrow^*_{\pi_a} Q$ hold with $Q = \nu\vec{a}.(A(\vec{b}) \mid Q')$ for some $\vec{b}$ and Q'?

Example 1.1 Let us consider the following equations:

$Init(a) = (\nu p)(\overline{a}p \mid Wait(p))$,

$Wait(p) = p(x).EndI(p, x)$,

$Resp(a) = a(y).(\nu ok)(\overline{y}ok \mid EndR(y, ok))$.

Given $P = (\nu c)(Init(c) \mid Resp(c))$, a possible reduction is as follows

$$\begin{aligned} P &= (\nu c, p)(\overline{c}p \mid Wait(p) \mid Resp(c)) \\ &\Rightarrow_{\pi_a} (\nu c, ok, p)(Wait(p) \mid \overline{p}\ ok \mid EndR(p, ok)) \\ &\Rightarrow_{\pi_a} (\nu c, ok, p)(EndI(p, ok) \mid EndR(p, ok)) \end{aligned}$$

This reduction describes a run of the protocol in which $Init$ and $Resp$ exchange the private channel name p along which $Resp$ sends an acknowledge to $Init$. If we add the equation

$$Start = (\nu c)(Init(c) \mid Resp(c) \mid Start)$$

then *Start* will generate an arbitrary number of *sessions* of our protocol.

1.2 The Specification Language MSR$_{NC}$

Let $\mathcal{V}$ be a set of variables. We call *name constraint* a conjunction $\varphi_1, \ldots, \varphi_n$ of atomic formulas of the shape *true*, $x > y$, $x = y$, $x \neq y$, or $x \geq y$ with $x, y \in \mathcal{V}$. The set of *solutions Sol* of a constraint φ consists of all evaluations from $\mathcal{V}$ to $\mathbb{Z}$ (*integer numbers*) that make φ true. A constraint φ is *satisfiable* whenever $Sol(\varphi) \neq \emptyset$.

Let $\mathcal{P}$ be a set of predicate symbols. An *atomic formula* $p(x_1, \ldots, x_n)$ is such that $p \in \mathcal{P}$, and $x_1, \ldots, x_n \in \mathcal{V}$. A *multiset* of atomic formulas is indicated as $A_1, \ldots, A_k$, where the symbol "," is an *associative-commutative* term constructor not occurring inside atomic formulas. We use "," instead of the symbol "|" used in [10] to avoid confusion with parallel composition in π_a. In the rest of the paper will use $\mathcal{M}$, $\mathcal{N}$, ... to denote *multisets* of atomic formulas, ϵ to denote the *empty multiset*, $\oplus$ to denote *multiset union* and $\ominus$ to denote *multiset difference*.

A *configuration* is a multiset of *ground atomic formulas*, i.e, atomic formulas were all variables are instantiated with *integer values*. An MSR$_{NC}$ *rule* has the form

$$A_1, \ldots, A_n \longrightarrow B_1, \ldots, B_m : \varphi$$

where $\mathcal{M} = A_1, \ \ldots, \ A_n$ and $\mathcal{M}' = B_1, \ \ldots, \ B_m$ are two (possibly empty) multisets of atomic formulas built on predicates in $\mathcal{P}$, and φ is a constraint such that $Var(\varphi) \subseteq Var(\mathcal{M}) \cup Var(\mathcal{M}')$, where $Var(F)$ is the set of free variables in the formula F. Equality constraints between variables can be implicitly defined by multiple occurrences of the same variable in a rule. The ground instances of an MSR$_{NC}$ rule are defined as

$$Inst(\mathcal{M} \longrightarrow \mathcal{M}' : \varphi) = \{\sigma(\mathcal{M}) \longrightarrow \sigma(\mathcal{M}') \mid \sigma \in Sol(\varphi)\}$$

where σ is extended in the natural way to multisets. The instances of a set of rules $\mathcal{R} = \{R_1, \ldots, R_n\}$ is defined as $Inst(\mathcal{R}) = Inst(R_1) \cup \ldots \cup Inst(R_n)$.

An MSR$_{NC}$ *specification* $\mathcal{S}$ is a tuple $\langle \mathcal{P}, \mathcal{I}, \mathcal{R} \rangle$, where $\mathcal{P}$ is a set of predicate symbols, $\mathcal{I}$ is a set of (*initial*) configurations, and $\mathcal{R}$ is a finite set of rules over $\mathcal{P}$. The *operational semantics* of $\mathcal{S}$ is defined via the rewriting relation $\Rightarrow_{\mathcal{R}}$ defined over configurations (i.e. ground multisets) as follows.

Given two configurations $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{M}_1 \Rightarrow_{\mathcal{R}} \mathcal{M}_2$ if and only if there exists a multiset of ground atomic formulas $\mathcal{Q}$ s.t. $\mathcal{M}_1 = \mathcal{N}_1 \oplus \mathcal{Q}$, $\mathcal{M}_2 = \mathcal{N}_2 \oplus \mathcal{Q}$, and $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ is in $Inst(\mathcal{R})$. A configuration $\mathcal{M}$ is *reachable* if there exists $\mathcal{M}_0 \in \mathcal{I}$ such that $\mathcal{M}_0 \stackrel{*}{\Rightarrow}_{\mathcal{R}} \mathcal{M}$, where $\stackrel{*}{\Rightarrow}_{\mathcal{R}}$ is the transitive closure of $\Rightarrow_{\mathcal{R}}$.

2 From π_a to MSR$_{NC}$

In this section we define an encoding of infinite-state asynchronous π-calculus specifications into MSR$_{NC}$. In this preliminary work we will restrict ourselves to *closed* π_a specifications and configurations. Specifically, we will consider normalised equations of the form $A(\vec{x}) = a(\vec{u}).(\nu\vec{v})R$ such that $Fn(R) \subseteq \{\vec{x}, \vec{u}, \vec{v}\}$, and configurations $(\nu\vec{v}).Q$ such that $Fn(Q) \subseteq \{\vec{v}\}$, i.e., we assume that all names with scope over different equations already occur in the quantifier associated to the initial configuration. Closed specifications and configurations present all the features of π_a we are interested in (fresh name generation, unbound parallelism, name and process mobility).

The encoding of closed specifications is defined as follows. Names are encoded as integer values. Relations over names are symbolically represented as constraints. We first encode an input action $\overline{a}\vec{x}$ and an output action $a(\vec{x})$ as the atomic formula $m(a, \vec{x})$, where $a, x_1, \ldots, x_n$ are free variables. Input messages will occur in the left-hand side of an MSR$_{NC}$ rule encoding a process definition, whereas output messages will occur in its right-hand side. Then, we encode a *process identifier* A using a predicate symbol p_A taking as arguments as many variables as the parameters in its defining equation. Finally, we use an atomic formula $new(f)$ to keep track of fresh values (i.e. to separate used and unused names). We will explain its meaning in few lines.

Let us consider the initial configuration P defined as

$$(\nu\vec{v})(\overline{a_1}\vec{y}_1 \mid \ldots \mid \overline{a}_n\vec{y}_n \mid A_1(\vec{w}_1) \ \ldots \mid A_m(\vec{w}_m))$$

The encoding of P is defined via the MSR$_{NC}$ rule $P^\bullet$ defined as

$$\begin{aligned}
&init, new(f) \longrightarrow \\
&\quad m(a_1, \vec{y}_1), \ldots, m(a_n, \vec{y}_n), p_{A_1}(\vec{w}_1), \ldots, p_{A_m}(\vec{w}_m), new(f') \ : \\
&\qquad\qquad f' > v_1, v_1 > v_2, \ldots, v_{r-1} > v_r, v_r > f.
\end{aligned}$$

The constraint over $f, f', \vec{v}$ ensures that the names in $\vec{v}$ are distinct each other, and that the global memory $new(f)$ contains a name strictly greater than all used names.

Let us consider now a *normalised* parametric equation D defined as

$$A(\vec{x}) = a(\vec{u}).(\nu\vec{v})(\overline{a}_1\vec{y}_1 \mid \ldots \mid \overline{a}_n\vec{y}_n \mid A_1(\vec{w}_1) \mid \ldots \mid A_m(\vec{w}_m))$$

The encoding of D is defined via the MSR$_{NC}$ rule $D^\bullet$ defined as

$$\begin{array}{l} p_A(\vec{x}), m(a, \vec{u}), new(f) \ \longrightarrow \\ \qquad m(a_1, \vec{y}_1), \ldots, m(a_n, \vec{y}_n), p_{A_1}(\vec{w}_1), \ldots p_{A_m}(\vec{w}_m), new(f') \ : \\ \qquad\qquad f' > v_1, v_1 > v_2, \ldots, v_{r-1} > v_r, v_r > f \end{array}$$

The constraint on $\vec{v}, f, f'$ ensures the freshness of the names in $\vec{v}$. Note that, if D has no generation of fresh values (i.e. $r = 0$), then we can simplify the rule by removing $new(f)$ and $new(f')$ from the left- and right-hand side, respectively.

Example 2.1 The MSR$_{NC}$ encoding of the equations of Example 1.1 is defined as follows (for brevity, we write $p_{Init}, \ldots$ as $init, \ldots$)

$$init, new(f) \rightarrow \ init(c), resp(c), new(f') \ : \ f' > c, c > f.$$

$$start, new(f) \rightarrow \ start, init(c), resp(c), new(f') \ : \ f' > c, c > f.$$

$$init(a), new(f) \ \rightarrow \ m(a, p), wait(p), new(f') : \ f' > p, p > f.$$

$$wait(p), m(p, x) \ \rightarrow \ endI(p, x) \ : \ true.$$

$$resp(a), m(a, y), new(f) \ \rightarrow \ m(y, ok), endR(y, ok), new(f') \ : \ f' > ok, ok > f.$$

Now, let P be the π_a configuration

$$(\nu\vec{v})(\overline{a_1}\vec{y}_1 \mid \ldots \mid \overline{a}_n\vec{y}_n \mid A_1(\vec{w}_1) \mid \ldots \mid A_m(\vec{w}_m))$$

Let η be an *injective mapping* from $\vec{v}$ to $\mathbb{Z}$ and let $\eta(\vec{v})$ denote $\eta(v_1), \ldots, \eta(v_n)$. We define $P^\bullet(\eta, N)$ as the MSR$_{NC}$ configuration

$$m(\eta(a_1), \eta(\vec{y}_1)), \ldots, m(\eta(a_n), \eta(\vec{y}_n)), p_{A_1}(\eta(\vec{w}_1)), \ldots, p_{A_m}(\eta(\vec{w}_m)), new(N)$$

where N is an integer strictly greater than $\eta(v_1), \ldots, \eta(v_r)$. Let $\eta : \vec{v} \rightsquigarrow \mathbb{Z}$, $\eta' : \vec{v'} \rightsquigarrow \mathbb{Z}$, and $\{\vec{v}\} \subseteq \{\vec{v'}\}$, then we define $\eta \leq \eta'$ if $\eta(\vec{v}) = \eta'(\vec{v})$. The adequacy of the encoding is established via the following propositions.

Proposition 2.2 *Let $\mathcal{E}$ be a set of closed normalised equations, and P_i be a closed configuration for $i : 1, \ldots, k$ such that $P_1 \Rightarrow_{\pi_a} \ldots \Rightarrow_{\pi_a} P_k$. Then, there exists $\eta_1 \leq \ldots \leq \eta_k$, and $N_1 \leq \ldots \leq N_k$ such that $P_1^\bullet(\eta_1, N_1) \Rightarrow_{\mathcal{E}^\bullet} \ldots \Rightarrow_{\mathcal{E}^\bullet} P_k^\bullet(\eta_k, N_k)$.*

Proof. The proof is by induction on the length k of the derivation, the interesting case being the inductive step in which $k \geq 1$. Suppose that $P_1 \Rightarrow_{\pi_a}$

$\ldots \Rightarrow_{\pi_a} P_k$ and that there exist $\eta_1 \leq \eta_2 \ldots \leq \eta_k$, and $N_1 \leq N_2 \ldots \leq N_k$ such that $P_1^\bullet(\eta_1, N_1) \Rightarrow_{\mathcal{E}^\bullet} \ldots \Rightarrow_{\mathcal{E}^\bullet} P_k^\bullet(\eta_k, N_k)$. Let P_k be the configuration

$$(\nu\vec{w})\ (\overline{c}\vec{z} \mid A(\vec{b}) \mid Q)$$

Suppose there exists a normalised equation

$$A(\vec{x}) = a(\vec{u}).(\nu\vec{v})R$$

and a substitution $\sigma = [\vec{x} \mapsto \vec{b}, \vec{u} \mapsto \vec{z}]$ such that $\hat{\sigma}(a) = c$. Then, $P_k \Rightarrow_{\pi_a} P_{k+1}$ where P_{k+1} is the configuration

$$(\nu\vec{w}, \vec{u})(\sigma(R) \mid Q)$$

By definition of the encoding, $P_k^\bullet(\eta_k, N_k)$ is the MSR$_{NC}$ configuration

$$m(\eta_k(c), \eta_k(\vec{z})), p_A(\eta_k(\vec{b})), new(N_k), Q'$$

where $\eta_k(c) < N_k$, and $d < N_k$ for any $d \in \{\eta_k(\vec{z}), \eta_k(\vec{b})\}$, and Q' is the encoding of the remaining part of the configuration Q. Furthermore, $D^\bullet$ is the rule

$$p_A(\vec{x}), m(a, \vec{u}), new(f) \longrightarrow R', new(f') : f' > v_1, \ldots, v_r > f.$$

where R' is the multiset corresponding to the encoding of the body of the equation. Let γ be a solution for $f' > v_1, \ldots, v_r > f$ such that $\gamma(a) = \eta_k(c)$, $\gamma(\vec{x}) = \eta_k(\vec{b})$, and $\gamma(\vec{u}) = \eta_k(\vec{z})$, and $\gamma(f) = N_k$. Furthermore, let η_{k+1} be defined in such a way that $\eta_k \leq \eta_{k+1}$ and $\eta_{k+1}(v_i) = \gamma(v_i)$ for $i : 1, \ldots, r$ and let $N_{k+1} = \gamma(f')$. Then,

$$p_A(\eta_k(\vec{b})), m(\hat{\eta_k}(c), \eta_k(\vec{z})), new(N_k) \longrightarrow \gamma(R'), new(N_{k+1}) \in Inst(D^\bullet).$$

where $\hat{\eta_k}$, $\hat{\gamma}$ represent the natural extensions of η_k and γ to (multiset of) terms. Furthermore, $P_{k+1}^\bullet(\eta_{k+1}, N_{k+1})$ is the MSR$_{NC}$ configuration

$$new(N_{k+1}), \hat{\eta}_{k+1}(Q'), \hat{\eta}_{k+1}(R)$$

By definition of $\Rightarrow_{\mathcal{E}^\bullet}$, it follows then that $P_k^\bullet(\eta_k, N_k) \Rightarrow_{\mathcal{E}^\bullet} P_{k+1}^\bullet(\eta_{k+1}, N_{k+1})$.$\Box$

Proposition 2.3 *Let $\mathcal{E}$ be a set of closed normalised equations and $\mathcal{E}^\bullet$ its MSR$_{NC}$ encoding, P_1 an initial closed configuration, and $\mathcal{M}_1 = P_1^\bullet(\eta_1, N_1)$ for some η_1 and N_1. If $\mathcal{M}_1 \Rightarrow_{\mathcal{E}^\bullet} \ldots \Rightarrow_{\mathcal{E}^\bullet} \mathcal{M}_k$, then there exists $P_2, \ldots, P_k$, $\eta_2 \leq \ldots \leq \eta_k$, and $N_2 \leq N_3 \leq \ldots \leq N_k$ such that $\eta_1 \leq \eta_2$, $N_1 \leq N_2$, $P_1 \Rightarrow_{\pi_a} \ldots \Rightarrow_{\pi_a} P_k$, and $\mathcal{M}_i = P_i^\bullet(\eta_i, N_i)$ for $i : 2, \ldots, k$.*

Proof. The proof is by induction on the length k of the derivation and follows a schema similar to the proof of the previous proposition. □

3 A General Procedure for Control Reachability

Control reachability is undecidable for generic specifications in asynchronous π-calculus, while its decidable for restricted fragments that can be embedded into Petri Nets (with transfer) [4]. However, the encoding described in the previous sections allows us to tackle this problem in its more general form via the *symbolic model checking* procedure we defined for MSR_{NC} in [10]. For studying the *control reachability problem* for specifications in asynchronous π-calculus we are interested in finitely representing configurations with an *arbitrary* number of names and processes. We will achieve this goal by resorting to the MSR_{NC} encoding of π-calculus configurations. Specifically, we introduce a symbolic representation of *upward closed* sets of configurations, called *constrained configuration.* A π_a *constrained configuration* is a formula

$$m(a_1, \vec{y}_1), \ldots, m(a_n, \vec{y}_n), p_{A_1}(\vec{w}_1), \ldots, p_{A_m}(\vec{w}_m), new(f) \ : \ \varphi \qquad (1)$$

defined over the set of variables $V = \{f, a_1, \ldots, a_n, \vec{y}_1, \ldots, \vec{y}_n, \vec{w}_1, \ldots, \vec{w}_m\}$ such that φ is an NC constraints over V, and $\varphi, f > x$ is satisfiable for *any* $x \in V$ $x \neq f$ (in every reachable configuration $new(f)$ separates used from unused names). The *denotation* of a set $\mathbf{S}$ of π_a constrained configurations is the *upward closure* of the *ground instances* of its elements, namely

$$[\![\mathbf{S}]\!] = \{\mathcal{N} \mid \mathcal{M} \preccurlyeq \mathcal{N},\ \mathcal{M} \in Inst(M),\ M \in \mathbf{S}\}$$

where $\preccurlyeq$ is *multiset inclusion*, and the operator $Inst$ is defined as

$$Inst(\mathcal{M} : \varphi) = \{\sigma(\mathcal{M}) \mid \sigma \in Sol(\varphi)\}.$$

Thus, a π_a constrained configuration like (1) represents the set of π_a-calculus configurations of the shape

$$(\nu\vec{v})(\xi(\overline{a_1})\xi(\vec{y}_1) \mid \ \ldots \ \mid \xi(\overline{a}_n)\xi(\vec{y}_n) \mid A_1(\xi(\vec{w}_1)) \mid \ \ldots \ \mid A_m(\xi(\vec{w}_m)) \mid Q)$$

where ξ is obtained by composing a solution σ for φ with an injective (possibly not surjective) mapping from $\mathbb{Z}$ to the set of names $\vec{v}$, and Q is any pool of messages and processes defined over a set of names containing $\vec{v}$.

This symbolic representation allows us to reason on *infinite sets* of configurations, thus forgetting about the actual number or processes/messages of a given run. Furthermore, the use of first order terms allows us to symbolically

$$\mathbf{Pre}_{\mathcal{R}}(\mathbf{S}) = \{\ \mathcal{A} \oplus \mathcal{N} \oplus \{new(x)\}\ :\ \xi\ \mid\ \text{cond (1-7) listed below holds}\ \}$$

(1) $\mathcal{A} \oplus \{new(x)\} \longrightarrow \mathcal{B} \oplus \{new(x')\} : \psi\ \in\ \mathcal{R}$,

(2) $\mathcal{M} \oplus \{new(y)\} : \varphi\ \in\ \mathbf{S}$,

(3) $\mathcal{B}' \preccurlyeq \mathcal{B},\ \mathcal{M}' \preccurlyeq \mathcal{M},\ \mathcal{N} = \mathcal{M} \ominus \mathcal{M}'$,

(4) $\sigma = mgu(\mathcal{M}', \mathcal{B}')$,

(5) $\gamma = \bigwedge_{w \in Var(\mathcal{A}\ \oplus \mathcal{N})} x > w$,

(6) $\xi = \exists x', y, \vec{z}.\ (\sigma \wedge \varphi \wedge \psi \wedge \gamma)$ is satisfiable

(7) $\vec{z} = Var(\sigma \wedge \varphi \wedge \psi) \setminus (Var(\mathcal{A}\ \oplus \mathcal{N}) \cup \{x\})$.

Fig. 1. Symbolic Predecessor Operator for Logical Encoding of π_a

represent an infinite number of different instances of the same collection of processes/messages. Constraints define the *relationship* between the data of different processes/messages.

3.1 Symbolic State Exploration for π_a

We can now define a *symbolic backward reachability* procedure that computes all predecessor states of a given set of π_a constrained configurations with respect to $\mathcal{E}^{\bullet}$. The procedure is based on a breadth-first visit of the infinite state space of the MSR_{NC} specification resulting from the encoding presented in the previous sections. The search is defined on the basis of a *symbolic predecessor operator* and on an entailment relation (over constrained configurations) both formally defined in [10].

To briefly explain the idea underlying the procedure given in [10], in the rest of this section we will present a specialization of the symbolic predecessor operator to the class of MSR_{NC} specification resulting from the encoding of π_a processes.

Let us first recall some definitions. Given two (multisets of) atomic formulas with distinct free variables t and t', a *unifier* for t and t' is a substitution σ such that $\sigma(t) = \sigma(t')$. The *most general unifier* $mgu(t, t')$ is the *idempotent* substitution σ such that any other unifier γ can be obtained from σ as $\gamma = \sigma \circ \eta$ for some substitution η; the most general unifier always exists and it is unique. In our settings unification might give rise to new bindings for integer variables. An mgu σ can also be viewed (and used) as an NC constraint of the shape of a conjunction of equalities $x = y$ for some variables x, y.

Let $\mathbf{S}$ be a set of π_a constrained configurations with distinct variables each other. The symbolic predecessor operator for an encoding in MSR_{NC} $\mathcal{R} = \mathcal{E}^{\bullet}$ of a specification in the asynchronous π-calculus $\mathcal{E}^{\bullet}$ is defined in Fig. 1. In the definition of Fig. 1 we combine *unification* (via the calculation of the most general unifier σ) and *constraint solving* (via satisfiability test and variable elimination); σ is needed to remember constraints on integer variables introduced via term unification. Condition $3-4$ in Fig. 1 ensures the existence of a common pool of messages and processes shared between the *right-hand side* of a rule and a π_a constrained configurations in $\mathbf{S}$. Condition 5 in Fig. 1 allows us to prune all configurations that violate the *freshness* of generated names (this is specific to the π_a encoding). Condition 6 ensures that the selected common multisets agree on the data part (i.e. the conjunction of their constraints is satisfiable). Existential quantification is used to project away all variables (Cond. (7)) not needed in the symbolic pre-image. The symbolic operator $\mathbf{Pre}_{\mathcal{R}}$ returns a *set of* π_a *constrained configurations* such that

$$[\![\mathbf{Pre}_{\mathcal{R}}(\mathbf{S})]\!] = Pre_{\mathcal{R}}([\![\mathbf{S}]\!])$$

for any set of π_a constrained configurations $\mathbf{S}$. This result follows from the result proved in [11] for the symbolic predecessor operator associated to an MSR_{NC} specification. The specialized operator of Fig. 1 is presented here only for giving an intuition on how the general search technique for MSR_{NC}. works.

The symbolic model checking procedure resulting from iterating the application of $\mathbf{Pre}_{\mathcal{R}}$ can be used then to attack *control reachability* for unrestricted π_a specifications. Let P be the initial configuration and A be the process identifier we would like to reach. Then, we can run the symbolic backward search starting from the symbolic configuration $A(\vec{x}), new(f) \ : \ \varphi$. If the search terminates we have to check then if *init* belongs to the resulting fixpoint. Clearly, in the MSR_{NC} we can use search procedure to check generalization of this problems in which the target set of configurations is defined via π_a constrained configurations like $A_1(\vec{x}_k), \ldots, A_k(\vec{x}_k), new(f) \ : \ \varphi$ and φ expresses the relation over the names of the different processes.

Example 3.1 As an example, suppose we want to check that in Example 1.1, the initial configuration $start \mid new(f)$ always generates sessions that do not interfere with each other, i.e., we never reach configurations like

$$(\nu\ c, d, d')(EndI(c, d) \mid EndR(c, d') \mid \ \ldots)$$

where $d' \neq d$, i.e., the processes involved in a session always exchange both names. To check this property we can apply the symbolic backward analysis described above starting from the π_a configuration

$$endI(x, y), endR(x, z), new(f) \ : \ z \neq y, f > x, f > y, f > z$$

Using our CLP-based implementation, this computation terminates in 6s, after a 4 steps, computing 22 π_a constrained configurations. The resulting fixpoint does not contain the configuration $start, new(f) : true$. This proves our original specification correct for an arbitrary number of Init-Resp sessions.

4 Related and Future Work

To our knowledge the present paper is the first attemp of establishing a connection between the infinite-state verification techniques based on constraints [1,3,2,10,11] and calculi for expressing mobility of processes as the asynchronous π_a calculus. Our verification method generalizes the ideas proposed for Time Petri Nets in [1,3] to more general classes of concurrent systems that can be specified via multiset rewriting and constraints. In previous work (see e.g. the technical report [11]) we applied our framework to mutual-exclusion and *data consistency protocols* (e.g. cache coherence). Multiset rewriting over first order atomic formulas has been proposed for specifying security protocols by Cervesato et al. in [8].

As future work we plan to extend the encoding to specifications in full π calculus, and to study the possible impact of the presented relationship for finding new decidable verification problems for π_a and π specifications.

References

[1] P. A. Abdulla and B. Jonsson. Verifying Networks of Timed Processes. TACAS'98, p. 298–312, 1998.

[2] P. A. Abdulla and B. Jonsson. Channel Representations in Protocol Verification. CONCUR '01, p. 1–15, 2001.

[3] P. A. Abdulla and A. Nylén. Better is Better than Well: On Efficient Verification of Infinite-State Systems. LICS '00, p. 132–140, 2000.

[4] R. Amadio and C. Meyssonier. On the Decidability of the Control Reachability Problem in the Asynchronous π-calculus. To appear in Nordic Journal of Computing.

[5] T. Ball, S. Chaki, and S. K. Rajamani. Parameterized Verification of Multithreaded Software Libraries. TACAS '01, p. 158-173, 2001.

[6] G. Boudol. Asynchrony and the π-calculus. Rapport de recherche 1702, INRIA. Sophia-Antipolis, 1992.

[7] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular Model Checking. CAV'00, p. 403–418, 2000.

[8] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A Meta-notation for Protocol Analysis. In *Proc. CSFW'99*, p. 55–69, 1999.

[9] S. Chaki, S. K. Rajamani, and J. Rehof. Types as models: model checking message-passing programs. POPL'02, p. 45-57, 2002.

[10] G. Delzanno. An Assertional Language for Systems Parametric in Several Dimensions. VEPAS'01, *ENTCS* volume 50, issue 4, 2001.

[11] G. Delzanno. On the Automated Verification of Parameterized Concurrent Systems with Unbounded Local Data. TR-DISI, Università di Genova, July 2002. Available at http://www.disi.unige.it/person/DelzannoG/MSR.

[12] J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. *LICS '99*, p. 352-359, 1999.

[13] S. M. German and A. P. Sistla. Reasoning about Systems with Many Processes. *Journal of the ACM*, 39(3):675–735, 1992.

[14] A. D. Gordon. Notes on Nominal Calculi for Security and Mobility. FOSAD'00, p. 262-330, 2000.

[15] K. Honda, M. Tokoro. An Object Calculus for Asynchronous Communication. ECOOP 1991: 133-147

[16] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes. *Information and Computation*, 100:1–77, 1992.

[17] U. Montanari, and M. Pistore. Checking bisimilarity for finitary π-calculus. CONCUR '95, p. 42-56, 1995.

[18] P. Yang, C.R. Ramakrishnan, S.A. Smolka. A Logical Encoding of the pi-Calculus: Model Checking Mobile Processes Using Tabled Resolution. VMCAI '03, p. 116-131, 2003.

[19] B. Victor, F. Moller The Mobility Workbench: A Tool for the Pi-Calculus. CAV '94, p. 428-440, 1994.