

Polymorphic Systems with Arrays: Decidability and Undecidability

Ranko Lazić
University of Warwick, UK

Tom Newcomb Bill Roscoe
University of Oxford, UK

LSV, ENS Cachan, France
27 April 2004

Context

Model checking infinite-state systems is an active research area:

- decidability,
- complexity,
- abstraction,
- semi-algorithms,
- tools, etc.

Many systems are infinite-state because they have *parameters* with infinite ranges, e.g. number of parallel components, data type. Seek correctness *for all* instantiations of the parameters.

Classes of infinite-state systems

Counting abstraction represents a system with an arbitrary number of identical parallel components as a Petri net [German & Sistla 92].

For more than rendez-vous communications, extensions of Petri nets are used, e.g.:

- for broadcasts, transfer arcs;
- for partially non-blocking rendez-vous, non-blocking arcs.

Some other abstract models related to Petri nets:

- broadcast protocols [Emerson & Namjoshi 98];
- multi-set rewriting specifications [Delzanno 02].

Many decidability/undecidability results, e.g.:

- [Esparza 98]
- [Esparza, Finkel & Mayr 99]
- [Raskin & Van Begin 03]
- [Lomazova & Schnoebelen 00]
- [Delzanno 02]

UNITY-style syntax

Infinite-state systems often given using state variables, guards and assignments.

For example:

$$cache : (X \times Y) \rightarrow (Z \times Enum_3)$$

UNITY-like syntax can succinctly express systems with *several parameters*. Petri nets and related models either restricted to one or two dimensions (e.g. broadcast protocols, Petri nets with non-blocking arcs), or complex (e.g. nested Petri nets).

Relating the two kinds of systems non-trivial in general.

Organisation

1. Polymorphic systems with arrays: syntax, semantics.
2. Control-state reachability problems.
3. Undecidability results.
4. Decidability result.

Running example: Bully Algorithm [Garcia-Molina 82].

λ -calculus: types

$$B ::= X \mid B_1 \times \cdots \times B_n \mid B_1 + \cdots + B_{n \geq 1}$$

$$T ::= B \mid B \rightarrow B'$$

$$\llbracket X \rrbracket_\omega = \omega[X]$$

$$\llbracket B_1 \times \cdots \times B_n \rrbracket_\omega = \llbracket B_1 \rrbracket_\omega \times \cdots \times \llbracket B_n \rrbracket_\omega$$

$$\llbracket B_1 + \cdots + B_n \rrbracket_\omega = \{1\} \times \llbracket B_1 \rrbracket_\omega \cup \cdots \cup \{n\} \times \llbracket B_n \rrbracket_\omega$$

$$\llbracket B \rightarrow B' \rrbracket_\omega = (\llbracket B' \rrbracket_\omega)^{\llbracket B \rrbracket_\omega}$$

λ -calculus: terms

$$t ::= x \mid (t_1, \dots, t_n) \mid \pi_i(t) \mid \\ \iota_i^B(t) \mid \text{case } t \text{ of } x_1.t'_1 \text{ or } \dots \text{ or } x_n.t'_n \mid \\ \lambda x : B . t \mid t_1[t_2]$$

Well-typed term-in-context:

$$\Omega, \Gamma \vdash t : T$$

$$\begin{aligned}
\llbracket x \rrbracket_{\omega, \gamma} &= \gamma \llbracket x \rrbracket \\
\llbracket (t_1, \dots, t_n) \rrbracket_{\omega, \gamma} &= (\llbracket t_1 \rrbracket_{\omega, \gamma}, \dots, \llbracket t_n \rrbracket_{\omega, \gamma}) \\
\llbracket \pi_i(t) \rrbracket_{\omega, \gamma} &= \pi_i(\llbracket t \rrbracket_{\omega, \gamma}) \\
\llbracket \nu_i^B(t) \rrbracket_{\omega, \gamma} &= (i, \llbracket t \rrbracket_{\omega, \gamma}) \\
\llbracket \text{case } t \text{ of } x_1.t'_1 \text{ or } \dots \text{ or } x_n.t'_n \rrbracket_{\omega, \gamma} &= \llbracket t'_i \rrbracket_{\omega, \gamma} \{x_i \mapsto v\}, \text{ where } (i, v) = \llbracket t \rrbracket_{\omega, \gamma} \\
\llbracket \lambda x : B . t \rrbracket_{\omega, \gamma} &= \{v \mapsto \llbracket t \rrbracket_{\omega, \gamma} \{x \mapsto v\} \mid v \in \llbracket B \rrbracket_{\omega}\} \\
\llbracket t_1[t_2] \rrbracket_{\omega, \gamma} &= \llbracket t_1 \rrbracket_{\omega, \gamma}(\llbracket t_2 \rrbracket_{\omega, \gamma})
\end{aligned}$$

λ -calculus: some abbreviations

$Unit = \text{empty product}$

$Bool = Unit + Unit$

$Enum_n = \overbrace{Unit + \dots + Unit}^n$

Polymorphic systems with arrays: syntax

A PSA is a 5-tuple $(\Omega, \Gamma, \Theta, R, I)$:

parameters: (Ω, Γ) is a signature;

state variables: Θ is disjoint from Γ , and $(\Omega, \Gamma \cup \Theta)$ is a signature;

instructions: each $\rho \in R$ is of the form

$$\Phi : c \cdot \{x_1 := t_1, \dots, x_k := t_k\}$$

instantiations: I is a set of instantiations of (Ω, Γ) .

Example array operations

reset:

$$a := \lambda x : B \cdot t$$

copy:

$$a := a'$$

map:

$$a := \lambda x : B \cdot t[(a'_1[x], \dots, a'_n[x])]$$

multiple partial assign:

$$a := \lambda x : B \cdot \text{if } d_1 \text{ then } t_1 \text{ elseif } \dots d_n \text{ then } t_n \text{ else } a[x]$$

abbreviated as

$$a[x : d_1; \dots; d_n] := t_1; \dots; t_n$$

write:

$$a[x : x = t_1; \dots; x = t_n] := t'_1; \dots; t'_n$$

abbreviated as

$$a[t_1; \dots; t_n] := t'_1; \dots; t'_n$$

cross-section:

$$a[x : (\pi_1(x) = t)] := t'$$

choose:

$$\langle a' : B \rightarrow B' \rangle : true \cdot \{a := a'\}$$

Polymorphic systems with arrays: semantics

The semantics of a PSA $(\Omega, \Gamma, \Theta, R, I)$ is the transition system (S, \rightarrow) :

states:

$$S = \{(\omega, \gamma, \theta) \mid (\omega, \gamma) \in I \wedge \theta \in \llbracket \Theta \rrbracket_{\omega}\}$$

transitions: $(\omega, \gamma, \theta) \rightarrow (\omega', \gamma', \theta')$ iff $\omega' = \omega$, $\gamma' = \gamma$, and there exists $\rho \in R$ which can produce θ' from θ .

More precisely, as ρ is of the form

$$\Phi : c \cdot \{x_1 := t_1, \dots, x_k := t_k\}$$

there exists $\phi \in \llbracket \Phi \rrbracket_\omega$ such that $\llbracket c \rrbracket_{\omega, \gamma \theta \phi} = tt$, and:

- $\theta' \llbracket x_i \rrbracket = \llbracket t_i \rrbracket_{\omega, \gamma \theta \phi}$ for each i ;
- $\theta' \llbracket x' \rrbracket = \theta \llbracket x' \rrbracket$ for all $x' \notin \{x_1, \dots, x_k\}$.

Example: Bully Algorithm

[Garcia-Molina 82]

Leader election in a distributed system.

Process identifiers linearly ordered.

1	2	3	4	\rightarrow^*
<i>Run</i>	<i>Run</i>	<i>Run</i>	<i>Coord</i>	fail (4)
<i>Run</i>	<i>Run</i>	<i>Run</i>	<i>Fld</i>	tock*
<i>Elect</i>	<i>Elect</i>	<i>Elect</i>	<i>Fld</i>	signal (2)
<i>Await</i>	<i>Elect</i>	<i>Elect</i>	<i>Fld</i>	tock* signal (3)
<i>Await</i>	<i>Await</i>	<i>Elect</i>	<i>Fld</i>	tock* fail (3)
<i>Await</i>	<i>Await</i>	<i>Fld</i>	<i>Fld</i>	tock*
<i>Coord</i>	<i>Elect</i>	<i>Fld</i>	<i>Fld</i>	signal (2) tock*
<i>Await</i>	<i>Coord</i>	<i>Fld</i>	<i>Fld</i>	signal (2)
<i>Run</i>	<i>Coord</i>	<i>Fld</i>	<i>Fld</i>	

Signature:

$$(\{X\}, \langle \leq_X: X \times X \rightarrow Bool \rangle)$$

Instantiations:

$$(X \mapsto \hat{k} = \{1, \dots, k\}, \leq_X \mapsto \leq_{\hat{k}})$$

State variables:

$$a : X \rightarrow (\{Elect, Coord, Await, Run, Fld\} \times \{1, \dots, T_S\} \times \{1, \dots, \max\{T_E, T_A, T_R\}\})$$

Some abbreviations: $a[t].m$, $a[t].c$, $a[t].c'$.

Instructions:

tock $\langle \rangle : true$.

$a := \lambda x : X$.

if $a[x].m \neq Fld \wedge a[x].c = T_S$ *then* $(Fld, 1, 1)$

elseif $a[x].m = Elect \wedge a[x].c' = T_E$ *then* $(Coord, a[x].c + 1, 1)$

elseif $a[x].m = Await \wedge a[x].c' = T_A$ *then* $(Elect, a[x].c + 1, 1)$

elseif $a[x].m = Run \wedge a[x].c' = T_R$ *then* $(Elect, a[x].c + 1, 1)$

elseif $a[x].m \neq Fld \wedge a[x].m \neq Coord$

then $(a[x].m, a[x].c + 1, a[x].c' + 1)$

elseif $a[x].m = Coord$ *then* $(a[x].m, a[x].c + 1, a[x].c')$

else $a[x]$

signal $\langle x : X \rangle : a[x].m \neq Fld.$

$a := \lambda x' : X. \text{ if } x' = x \text{ then } (a[x].m, 1, a[x].c')$

$\text{elseif } x' < x \wedge a[x].m \neq Coord \wedge$

$a[x'].m \in \{Elect, Coord\} \text{ then } (Await, a[x'].c, 1)$

$\text{elseif } x' < x \wedge a[x].m = Coord \wedge a[x'].m \neq Fld$

$\text{then } (Run, a[x'].c, 1)$

$\text{else } a[x']$

fail $\langle x : X \rangle : a[x].m \neq Fld \cdot a[x] := (Fld, 1, 1)$

revive $\langle x : X \rangle : a[x].m = Fld \cdot a[x] := (Elect, 1, 1)$

Initialised control-state reachability

Suppose we have a PSA $(\Omega, \Gamma, \Theta, R, I)$ with:

- a state variable $b : Enum_n$,
- $i, j \in \{1, \dots, n\}$, and
- for each array state variable $a : B \rightarrow B'$, a term $\Omega, \Gamma \Theta_{bas} \vdash t_a : B'$.

To decide whether there exists a sequence of transitions from a state satisfying

$$b = e_i \wedge \bigwedge_{a: B \rightarrow B' \in \Theta} \forall x : B \cdot a[x] = t_a$$

to a state satisfying

$$b = e_j$$

Uninitialised control-state reachability

Suppose we have a PSA $(\Omega, \Gamma, \Theta, R, I)$ with:

- a state variable $b : Enum_n$, and
- $i, j \in \{1, \dots, n\}$.

To decide whether there exists a sequence of transitions from a state satisfying

$$b = e_i$$

to a state satisfying

$$b = e_j$$

Example: Bully Algorithm

- *There are never two distinct processes in Coord mode.*

We add a state variable $b : \{0, 1\}$, and an instruction

$$\langle x : X, x' : X \rangle : x \neq x' \wedge a[x].m = \text{Coord} \wedge a[x'].m = \text{Coord} \cdot b := 1$$

The check is whether, from a state in which

$$b = 0 \wedge \forall x : X \cdot a[x] = (\text{Elect}, 1, 1)$$

the system can reach a state in which

$$b = 1$$

- *A process cannot continuously be Run since receiving a signal from a Coord until receiving a signal from a Coord whose identifier is smaller than that of the previous one.*
- *There is never a Coord process and a Run process with a greater identifier.*

We add a state variable $b : \{0, 1\}$, and an instruction

$$\langle x : X, x' : X \rangle : x < x' \wedge a[x].m = \text{Coord} \wedge a[x'].m = \text{Run} \cdot b := 1$$

The check is as in the first example.

Theorem 1

Initialised CSR is undecidable for each of the following classes of PSAs:

$X \times X$ -to-Bool • signature is $(\{X\}, \langle =_X : X \times X \rightarrow Bool \rangle)$;

- only one array state variable, of type $X \times X \rightarrow Bool$;
- no instruction parameters which are arrays, and each array assignment is a write;
- instantiations are $(X \mapsto \hat{k}, =_X \mapsto =_{\hat{k}})$.

$X \times Y$ -to-Bool • signature is

$$(\{X, Y\}, \langle =_X: X \times X \rightarrow Bool, =_Y: Y \times Y \rightarrow Bool \rangle)$$

- only one array state variable, of type $X \times Y \rightarrow Bool$;
- no instruction parameters which are arrays, and each array assignment is a write;
- instantiations are

$$(X \mapsto \hat{k}, Y \mapsto \hat{l}, \\ =_X \mapsto =_{\hat{k}}, =_Y \mapsto =_{\hat{l}})$$

X -to- Y, Z • signature is

$$\langle \{X, Y, Z\}, \\ \langle =_X: X \times X \rightarrow Bool, =_Y: Y \times Y \rightarrow Bool, =_Z: Z \times Z \rightarrow Bool \rangle \rangle$$

- only two array state variables, of types $X \rightarrow Y$ and $X \rightarrow Z$;
- no instruction parameters which are arrays, and each array assignment is a write;
- instantiations are

$$(X \mapsto \hat{k}, Y \mapsto \hat{l}, Z \mapsto \hat{m}, \\ =_X \mapsto =_{\hat{k}}, =_Y \mapsto =_{\hat{l}}, =_Z \mapsto =_{\hat{m}})$$

X, \leq -to- Y • signature is

$$(\{X, Y\}, \langle \leq_X: X \times X \rightarrow Bool, =_Y: Y \times Y \rightarrow Bool \rangle)$$

- only one array state variable, of type $X \rightarrow Y$;
- no instruction parameters which are arrays, and each array assignment is a write;
- instantiations are

$$(X \mapsto \hat{k}, Y \mapsto \hat{l}, \\ \leq_X \mapsto \leq_{\hat{k}}, =_Y \mapsto =_{\hat{l}})$$

Corollary

For classes of PSAs obtained by extending the classes above to allow resets of arrays, uninitialised CSR is undecidable.

Proof of Theorem 1

By reducing from *location reachability* for two-counter machines: given a 2CM and a location L_j , to decide whether a configuration with location L_j is reachable from $(L_1, 0, 0)$.

$X \times X$ -to-Bool

$$c_1 = 3$$

$$c_2 = 1$$

			x'_1			x'_2	
x_1		t					1
				t			2
							3
			t				4
x_2						t	5
							6
	1	2	3	4	5	6	

$X \times Y$ -to-Bool

$$c_1 = 2$$

$$c_2 = 1$$

	y_1		y'_1	y_2		y'_2
x_1	t					
x_2				t		
	t	t				
x'_1		t	t			
x'_2				t		t

X-to-Y, Z

$$c_1 = 2$$

$$c_2 = 1$$

	x_1				x'_1		x_2			x'_2	
	y_1^1	y_2^1	y_1^1	y_2^1			y_1^2	y_2^2			
		z_2^1	z_1^1	z_1^1	z_2^1			z_1^2		z_1^2	

X, \leq -to- Y

$$c_1 = 3$$

$$c_2 = 1$$

	x_1							x'_1	x''_1		x_2	x'_2	x''_2
	y_1		y_2	y_1		y_3		y_4	y_3		y_1	y_2	y_1

Theorem 2

Initialised and uninitialised CSR problems are decidable for the class X, \leq -to-Enum of PSAs:

- signature is $(\{X\}, \langle \leq_X: X \times X \rightarrow Bool \rangle)$;
- the type of any array state variable, and of any array instruction parameter, is of the form $X \rightarrow Enum_m$;
- instantiations are $(X \mapsto \hat{k}, \leq_X \mapsto \leq_{\hat{k}})$.

Proof of Theorem 2

By reducing to whether a monadic MSR(NC) specification can reach the upward closure of a finite set of constrained configurations.

The latter problem is decidable, and a decision procedure has been implemented [Delzanno 02].

We have an instance of the initialised or uninitialised CSR problem, which is for a PSA $(\Omega, \Gamma, \Theta, R, I)$ in the class X, \leq -to-Enum.

By properties of the λ -calculus, we can assume:

- Θ is of the form

$$\langle b : Enum_n, x_1 : X, \dots, x_l : X, a : X \rightarrow Enum_m \rangle$$

- the parameters of any $\rho \in R$ are of the form

$$\langle x_{l+1} : X, \dots, x_{l+l'} : X, a' : X \rightarrow Enum_{m'} \rangle$$

We construct a monadic MSR(NC) specification $(\mathcal{P}, \text{NC}, \mathcal{I}, \mathcal{R})$. Let \mathcal{P} consist of:

- nullary predicate symbols z, nz, b_1, \dots, b_n ;
- unary predicate symbols x_1, \dots, x_l ;
- unary predicate symbols $aa'_{i,j}$ for $i \in \{1, \dots, m\}, j \in \{0, 1, \dots, m'\}$.

NC is the system of name constraints:

$$\varphi ::= \text{false} \mid \text{true} \mid x = x' \mid x < x' \mid \varphi \wedge \varphi'$$

NC constraints are interpreted over the integers \mathbb{Z} .

For any state (ω, γ, θ) of the PSA $(\Omega, \Gamma, \Theta, R, I)$, where $\omega = \{X \mapsto \hat{k}\}$ and $\gamma = \{\leq_X \mapsto \leq_{\hat{k}}\}$, let

$$F(\omega, \gamma, \theta) = z \mid \mathbf{b}_{\theta[[b]]} \mid x_1(\theta[[x_1]]) \mid \cdots \mid x_l(\theta[[x_l]]) \mid \\ \mathbf{aa}'_{\theta[[a]](1),0}(1) \mid \cdots \mid \mathbf{aa}'_{\theta[[a]](k),0}(k)$$

The MSR(NC) specification $(\mathcal{P}, \text{NC}, \mathcal{I}, \mathcal{R})$ can reach a configuration \mathcal{M} with $z \in \mathcal{M}$ from $F(\omega, \gamma, \theta)$ if and only if $\mathcal{M} = F(\omega, \gamma, \theta')$ for some state $(\omega, \gamma, \theta')$ reachable from (ω, γ, θ) .