Beyond Regular Model Checking

Dana Fisman

joint work with

Amir Pnueli

Minerva Center of Verification of Reactive Systems
Department of Computer Science
The Weizmann Institute of Science

May 3, 2001

The problem:

Uniform Verification of Parameterized Systems

The problem:

Uniform Verification of Parameterized Systems

A Parameterized System consists of a family of systems $\mathcal{S}(N)$ for every natural $N > 0$.

$$\|_{i=1}^{N} P[i] :: \left(\left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: Non-Critical} \\ \ell_2\text{: request } x \\ \ell_3\text{: Critical} \\ \ell_4\text{: release } x \end{array} \right] \end{array} \right] \right)$$

Beyond Regular

The problem:

Uniform Verification of Parameterized Systems

A Parameterized System consists of a family of systems $\mathcal{S}(N)$ for every natural $N > 0$.

$$\|_{i=1}^{N} P[i] :: \left(\left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: Non-Critical} \\ \ell_2\text{: request } x \\ \ell_3\text{: Critical} \\ \ell_4\text{: release } x \end{array} \right] \end{array} \right] \right)$$

The System $\mathcal{S}(N)$ is obtained by instantiating the parameter.

$$\mathcal{S}(N) \quad \overbrace{\left(\left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: Non-Critical} \\ \ell_2\text{: request } x \\ \ell_3\text{: Critical} \\ \ell_4\text{: release } x \end{array} \right] \end{array} \right] \right)}^{P[1]} \| \overbrace{\left(\left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: Non-Critical} \\ \ell_2\text{: request } x \\ \ell_3\text{: Critical} \\ \ell_4\text{: release } x \end{array} \right] \end{array} \right] \right)}^{P[2]} \| \quad \cdots \quad \| \overbrace{\left(\left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: Non-Critical} \\ \ell_2\text{: request } x \\ \ell_3\text{: Critical} \\ \ell_4\text{: release } x \end{array} \right] \end{array} \right] \right)}^{P[N]}$$

Uniform Verification $\quad \forall N \; \mathcal{S}(N) \models \varphi$

Beyond Regular

The Parameterized Verification Problem

- The parameterized verification problem is undecidable [AK86]
- Approaches
  - Explicit Induction
    [SG92][EN96]
  - Network Invariants
    [WL89][KM95][HLR92][HLR97][KP00]
  - Abstractions and Approximations of Network Invariants
    [BCG86][SG89][CGJ95]
  - Abstractions
    [ID96][EN96]
  - Cutoffs
    [EK00][PRZ01]
  - Regular Model Checking
    [KMMPS97][WB98]

    \+ Completely Automatic

    − Limited to Systems which can be modeled by regular languages

Beyond Regular

Goal

uniform verification of
non-regular
parameterized systems

Beyond Regular Model Checking

Outline

- Modeling Systems by Languages
- The Peterson Algorithm
- Symbolic Model Checking and Regular Model Checking
- Adequacy of Classes of Languages for SMC
- We show 3 Methods Beyond Regular Model Checking

Modeling Systems by Languages

A *system* is a quadruple $(\Sigma, \chi, \Theta, \rho)$ consisting of the following components:

- Σ is a finite alphabet.
- $\chi \subseteq \Sigma^*$ is a language over Σ, denoting the set of *states*.
- $\Theta \subseteq \chi$ is a language over Σ, denoting the set of *initial states*.
- $\rho \subseteq \chi \times \chi$ is a bi-language over Σ, denoting the *transition relation*.

$$s = a_1 a_2 \ldots a_n$$

$$s' = b_1 b_2 \ldots b_m$$

$$\rho(s, s')$$

$$w = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \cdots \begin{bmatrix} a_n \\ b_n \end{bmatrix}$$

$$w \in \rho$$

Beyond Regular

Modeling Systems by Languages

- Finite-State Systems
- Parameterized Systems
- Systems with Unbounded Variables
- Parameterized Systems with Unbounded Variables

Modeling Parameterized Systems by Languages

The Token-String Example

Σ Local state: $\{0,1\}$ i.e. without / with a token

χ Global states: $\{0,1\}^+$ e.g. 0100

Θ Initial states: 10^*

ρ Transitions: $\left(\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}\right)^* \begin{bmatrix}1\\0\end{bmatrix} \begin{bmatrix}0\\1\end{bmatrix} \left(\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}\right)^*$

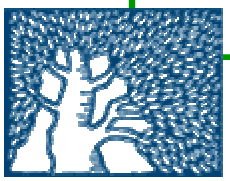
$X\ 1\ 0\ Y \rightarrow X\ 0\ 1\ Y$

Modeling Parameterized Systems with Unbounded Variables by Languages

N : natural initially $N > 1$
y : array $[1..N]$ of $[0..N-1]$ initially $y = 0$
s : array $[1..N-1]$ of $[1..N]$

$$\|_{i=1}^{N} P[i] :: \left[\begin{array}{l} t : \text{integer} \\ \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: Non-Critical} \\ \ell_2\text{: for } t := 1 \text{ to } N-1 \text{ do} \\ \quad \left[\begin{array}{l} \ell_3\text{: } (y[i], s[t]) := (t, i) \\ \ell_4\text{: await } s[t] \neq i \ \vee \ \forall j \neq i : y[j] < y[i] \end{array}\right] \\ \ell_5\text{: Critical} \\ \ell_6\text{: } y[i] := 0 \end{array}\right] \end{array}\right]$$

Peterson Algorithm for Mutual Exclusion among N Processes

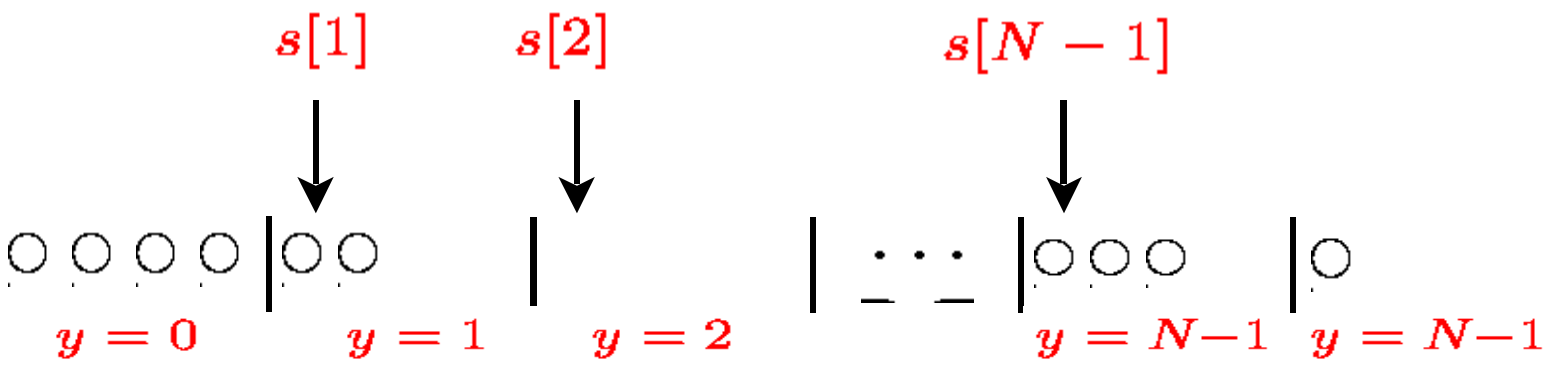


Beyond Regular

An Encoding for Peterson

We can encode a state of the system as a word of the form

○– processor

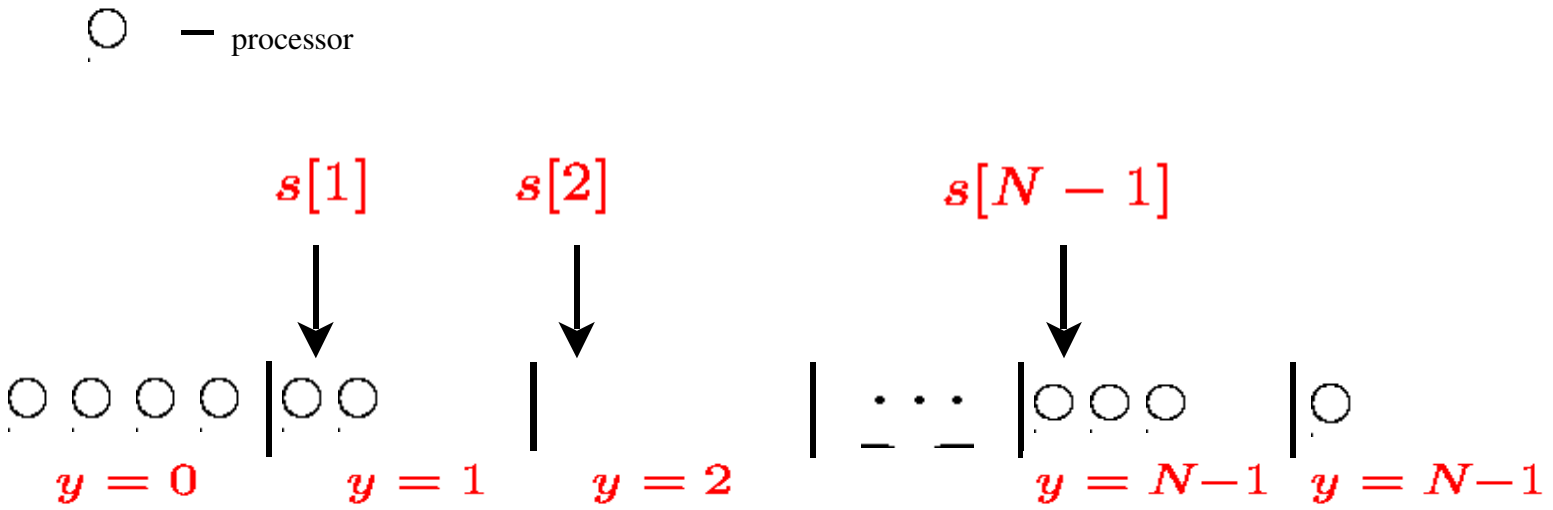


Critical

Beyond Regular

An Encoding for Peterson

We can encode a state of the system as a word of the form



Critical

Beyond Regular

An Encoding for Peterson

$$\chi = \left\{ w \in (\bigcirc + |)^* \ : \quad \sharp(|, w) = \sharp(\bigcirc, w) > 1 \right\}$$

$$\Theta = \left\{ \bigcirc^i |^i \ : \quad i > 1 \right\}$$

ρ_1	:	$u \bigcirc \mid v \mapsto u \mid \bigcirc v$	where $u \in \bigcirc^*, v \in (\mid + \bigcirc)^*$
ρ_2	:	$u \mid \bigcirc \mid v \mapsto u \mid \mid \bigcirc v$	where $u \in (\mid + \bigcirc)^*, v \in \mid^*$
ρ_3	:	$u \bigcirc \bigcirc \mid v \mapsto u \bigcirc \mid \bigcirc v$	where $u, v \in (\mid + \bigcirc)^*$
ρ_4	:	$u \bigcirc \mapsto \bigcirc u$	where $u \in (\mid + \bigcirc)^*$
ρ_{id}	:	$u \mapsto u$	where $u \in (\mid + \bigcirc)^*$

Beyond Regular

Another Encoding for Peterson

$\Sigma = \{0, 1, 2\}$ Empty / Exactly one / Two or more

$$\chi = \left\{ uavbw \;\middle|\; \begin{array}{l} u, v, w \in \Sigma^*, \quad a, b \in \Sigma, \\ |uavbw| \geq 3 \text{ and } a + b \geq 2 \end{array} \right\}$$

$$\Theta = \left\{ \; 20^i \;\middle|\; i \geq 2 \; \right\}$$

$$\rho_1 : \left\{ \begin{array}{lcl} 2\,0\,v & \mapsto & 2\,1\,v \\ 2\,1\,v & \mapsto & 2\,2\,v \\ 1\,0\,v & \mapsto & 0\,1\,v \\ 1\,1\,v & \mapsto & 0\,2\,v \\ 1\,2\,v & \mapsto & 0\,2\,v \end{array} \right.$$

$\rho_2 : \; u\,1\,0\,v \mapsto u\,0\,1\,v$ where $v \in 0^*$

$\rho_{id} : \; u \mapsto u$

$$\rho_3 : \left\{ \begin{array}{lcl|l} u\,2\,0\,v & \mapsto & u\,1\,1\,v & u\,2\,1\,v \\ u\,2\,1\,v & \mapsto & u\,1\,2\,v & u\,2\,2\,v \\ u\,2\,2\,v & \mapsto & u\,1\,2\,v & \end{array} \right.$$

$$\rho_4 : \left\{ \begin{array}{lcl|l} 0\,u\,2 & \mapsto & 1\,u\,1 & 1\,u\,2 \\ 0\,u\,1 & \mapsto & 1\,u\,0 & \\ 1\,u\,2 & \mapsto & 2\,u\,1 & 2\,u\,2 \\ 1\,u\,1 & \mapsto & 2\,u\,0 & \\ 2\,u\,1 & \mapsto & 2\,u\,0 & \\ 2\,u\,2 & \mapsto & 2\,u\,1 & 2\,u\,2 \end{array} \right.$$

Beyond Regular

Symbolic Model checking (SMC) [BCM+92]

Input: a system $\mathcal{S} = (\Sigma, \chi, \Theta, \rho)$
an invariance property φ

Question: Does $\mathcal{S} \models \varphi$?

Symbolic Model checking (SMC) [BCM+92]

Input: a system $\mathcal{S} = (\Sigma, \chi, \Theta, \rho)$
an invariance property φ

Question: Does $\mathcal{S} \models \varphi$?

Procedure ***Forward SMC***
let $\psi_0 := \Theta$
for $i = 0, 1, \ldots$ **repeat**
let $\psi_{i+1} := \psi_i \cup (\psi_i \diamond \rho)$
until $\psi_{i+1} = \psi_i$
return $\psi_i \cap \overline{\varphi} = \emptyset$

where $\psi \diamond \rho$ denotes ρ-successor of φ

$$\psi \diamond \rho = \left\{ s' \;\middle|\; \exists s \in \psi \text{ such that } \begin{bmatrix} s \\ s' \end{bmatrix} \in \rho \right\}$$

Procedure ***Backward SMC***
let $\psi_0 := \overline{\varphi}$
for $i = 0, 1, \ldots$ **repeat**
let $\psi_{i+1} := \psi_i \cup (\rho \diamond \psi_i)$
until $\psi_{i+1} = \psi_i$
return $\psi_i \cap \Theta = \emptyset$

where $\rho \diamond \psi$ denotes ρ-predecessor of φ

$$\rho \diamond \psi = \left\{ s \;\middle|\; \exists s' \in \psi \text{ such that } \begin{bmatrix} s \\ s' \end{bmatrix} \in \rho \right\}$$

Beyond Regular

Symbolic Model checking (SMC) [BCM+92]

- In symbolic model checking, some symbolic representation $\mathcal{C}$ is used to represent sets of states.
- The system and the property to be verified are represented as expressions in $\mathcal{C}$.
- The model checking procedures operate by manipulating sets of states (instead of individual states) through operations on the expressions of $\mathcal{C}$.
  - Finite-state systems
    - Symbolic representation: OBDDs
    - termination is guaranteed
  - Infinite-state systems -
    - requires symbolic representation
    - termination is not guaranteed.

Regular Model Checking

Application of symbolic model checking, where regular expressions are used as the (symbolic) assertional language for describing sets of states.

Forward exploration:

$$\begin{array}{lcll} \varphi_0 & = & \Theta & : \quad 10^* \\ \varphi_1 & = & \Theta + \Theta \diamond \rho & : \quad \varphi_0 + 010^* \\ \varphi_2 & = & \Theta + \Theta \diamond \rho + (\Theta \diamond \rho) \diamond \rho & : \quad \varphi_1 + 0010^* \\ & & \cdots & \end{array}$$

Computation doesn't converge!

Backward exploration:

$$\begin{array}{lcll} \varphi_0 & = & \overline{\varphi} & : \quad 0^* + (0+1)^*\ 1\ (0+1)^*\ 1\ (0+1)^* \\ \varphi_1 & = & \overline{\varphi} + \rho \diamond \overline{\varphi} & : \quad 0^* + (0+1)^*\ 1\ (0+1)^*\ 1\ (0+1)^* \end{array}$$

Computation terminates in one iteration step.

Adequacy of Classes of Languages for SMC

Beyond Regular

Adequacy of Classes of Languages for SMC

Let $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{A}$ be three classes of languages.

Let $A_\varphi, A_{Init} \in \mathcal{A}$, $M_{Init}, M_\varphi \in \mathcal{M}$, $R \in \mathcal{R}$ and $M_0, M_1, M_2, \ldots \in \mathcal{M}$.

Procedure *Forward MC*

$M_0 := M_{Init}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((M_i \times \Sigma^*) \cap R)\Downarrow_2$

until $M_{i+1} = M_i$

return $M_i \cap \overline{A_\varphi} = \emptyset$

Procedure *Backward MC*

$M_0 := \overline{M_\varphi}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((\Sigma^* \times M_i) \cap R)\Downarrow_1$

until $M_{i+1} = M_i$

return $M_i \cap A_{Init} = \emptyset$

ρ-**successor** of φ

$$\varphi \diamond \rho = \left\{ s' \;\middle|\; \exists s \in \psi \text{ such that } \begin{bmatrix} s \\ s' \end{bmatrix} \in \rho \right\}$$

$$\varphi \diamond \rho = ((\psi \times \Sigma^*) \cap \rho)\Downarrow_2$$

ρ-**predecessor** of φ

$$\rho \diamond \varphi = \left\{ s \;\middle|\; \exists s' \in \psi \text{ such that } \begin{bmatrix} s \\ s' \end{bmatrix} \in \rho \right\}$$

$$\rho \diamond \varphi = \rho \diamond \psi = ((\Sigma^* \times \psi) \cap \rho)\Downarrow_1$$

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \cdots \begin{bmatrix} a_n \\ b_n \end{bmatrix}$$

Beyond Regular

Adequacy of Classes of Languages for SMC

Let $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{A}$ be three classes of languages.

Let $A_\varphi, A_{Init} \in \mathcal{A}$, $M_{Init}, M_\varphi \in \mathcal{M}$, $R \in \mathcal{R}$ and $M_0, M_1, M_2, \ldots \in \mathcal{M}$.

Procedure *Forward MC*

$M_0 := M_{Init}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((M_i \times \Sigma^*) \cap R)\Downarrow_2$

until $M_{i+1} = M_i$

return $M_i \cap \overline{A_\varphi} = \emptyset$

Procedure *Backward MC*

$M_0 := \overline{M_\varphi}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((\Sigma^* \times M_i) \cap R)\Downarrow_1$

until $M_{i+1} = M_i$

return $M_i \cap A_{Init} = \emptyset$

ρ-**successor** of φ

$$\varphi \diamond \rho = \left\{ s' \;\middle|\; \exists s \in \psi \text{ such that } \begin{bmatrix} s \\ s' \end{bmatrix} \in \rho \right\}$$

$$\varphi \diamond \rho = ((\psi \times \Sigma^*) \cap \rho)\Downarrow_2$$

ρ-**predecessor** of φ

$$\rho \diamond \varphi = \left\{ s \;\middle|\; \exists s' \in \psi \text{ such that } \begin{bmatrix} s \\ s' \end{bmatrix} \in \rho \right\}$$

$$\rho \diamond \varphi = \rho \diamond \psi = ((\Sigma^* \times \psi) \cap \rho)\Downarrow_1$$

$b_1 \quad b_2 \quad b_3 \quad \cdots \quad b_n$

Beyond Regular

Adequacy of Classes of Languages for SMC

The classes $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{A}$ are adequate for SMC if:

1. $\mathcal{M}$ and $\mathcal{A}$ are adequate for specifying Θ and φ respectively, and $\mathcal{R}$ is adequate for representing ρ.
2. $\mathcal{A}$ is effectively closed under complementation.
3. $\mathcal{M}$ is effectively closed under lifting.
4. $\mathcal{M}$ is effectively closed under intersection with $\mathcal{R}$.
5. $\mathcal{M}$ is effectively closed under projection.
6. Either $\mathcal{M}$ is effectively closed under intersection with $\mathcal{A}$ and emptiness is decidable for $\mathcal{M}$, or $\mathcal{A}$ is effectively closed under intersection with $\mathcal{M}$ and emptiness is decidable for $\mathcal{A}$.
7. Equivalence of two languages in $\mathcal{M}$ is decidable.

Procedure *Forward MC*

$M_0 := M_{Init}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((M_i \times \Sigma^*) \cap R)\Downarrow_2$

until $M_{i+1} = M_i$

return $M_i \cap \overline{A_\varphi} = \emptyset$

Adequacy of Classes of Languages for SMC - Meeting the Requirements

1. $\mathcal{M}$- Θ, $\mathcal{A}$ - φ and $\mathcal{R}$ - ρ.
2. $\mathcal{A}$ - complementation.
3. $\mathcal{M}$ - lifting.
4. $\mathcal{M}$ - intersection with $\mathcal{R}$.
5. $\mathcal{M}$ - projection.
6. • $\mathcal{M}$ - intersection with $\mathcal{A}$ and emptiness
   • $\mathcal{A}$ - intersection with $\mathcal{M}$ and emptiness
7. $\mathcal{M}$ - Equivalence

Regular Model Checking

- $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{A}$ are taken to be class $\mathcal{L}_{FA}$.

Beyond Regular Model Checking

- $\mathcal{A}$ is the class $\mathcal{L}_{DPDA}$ and $\mathcal{M}$, $\mathcal{R}$ and are taken to be class $\mathcal{L}_{FA}$. ✓
- $\mathcal{A}$ is the class $\mathcal{L}_{PDA}$ and $\mathcal{M}$, $\mathcal{R}$ and are taken to be class $\mathcal{L}_{FA}$. ✓
- $\mathcal{M}$ is the class $\mathcal{L}_{DPDA}$ and $\mathcal{A}$, $\mathcal{R}$ and are taken to be class $\mathcal{L}_{FA}$. ✗ projection

Beyond Regular Model checking

overview of the 3 methods

- Method 1: Computing reachable set by regular means and **then** intersecting with a CFL.
- Method 2: Computing with cascade product $1\mathrm{DPDA} \circ \mathrm{DFA}$.
- Method 3: Computing with product $\mathrm{DPDA} \times \mathrm{DFA}$.

Method 1: Computing reachable set by regular means and then intersecting with a CFL.

Method 1: Computing reachable set by regular means and then intersecting with a CFL.

- Start with a regular language (M_φ / M_{Init}, R)
- Compute the regular fix point
- Intersect with a context-free language ($A_{Init} / \overline{A_\varphi}$)

Procedure *Forward MC*

$M_0 := M_{Init}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((M_i \times \Sigma^*) \cap R)\Downarrow_2$

until $M_{i+1} = M_i$

return $M_i \cap \overline{A_\varphi} = \emptyset$

Procedure *Backward MC*

$M_0 := \overline{M_\varphi}$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((\Sigma^* \times M_i) \cap R)\Downarrow_1$

until $M_{i+1} = M_i$

return $M_i \cap A_{Init} = \emptyset$

Beyond Regular

Method 1: Computing reachable set by regular means and then intersecting with a CFL.

Applying Regular Model Checking

Globals: $\chi = \left\{ uavbw \;\middle|\; \begin{array}{l} u,v,w \in \Sigma^*, \quad a,b \in \Sigma, \\ |uavbw| \geq 3 \text{ and } a+b \geq 2 \end{array} \right\}$

Initials: $\Theta = \left\{ \; 20^i \;\middle|\; i \geq 2 \; \right\}$

Reachables: $\psi = \chi \cap (\Theta \cup L_1 \cup L_2)$ where

$L_1 = \left\{ \; au2v \;\middle|\; a \in \{0,1,2\} \;\; u \in \{1,2\}^* \;,\; v \in 0^* \; \right\}$

$L_2 = \left\{ \; auv1w \;\middle|\; a \in \{0,1,2\} \;\; u \in \{1,2\}^* \;,\; v,w \in 0^* \; \right\}$

Property: $\varphi^c = \left\{ \; x2 \;\middle|\; x \in \{0,1,2\}^* \; \right\}$

Intersection: $\psi \cap \varphi^c = \left\{ \; au2 \;\middle|\; a \in \{0,1,2\} \;\; u \in \{1,2\}^* \; \right\} \neq \emptyset.$

Thus, we were unable to verify the mutual exclusion property for Peterson(N), using regular model checking.

Beyond Regular

Method 1: Computing reachable set by regular means and then intersecting with a CFL.

Applying SMC Using Context-Free Languages

Globals: $\widetilde{\chi} = \{\ w\ :\ \ \sharp(|, w) = \sharp(\bigcirc, w) \geq 2\ \}$

Initials: $\widetilde{\Theta} = \left\{\ \bigcirc^i|^i\ :\ \ i \geq 2\ \right\}$

Reachables: $\widetilde{\psi} = \widetilde{\chi} \cap (\widetilde{\Theta} \cup \widetilde{L_1} \cup \widetilde{L_2})$ where

$$\widetilde{L_1} = \bigcirc^*| \left(\bigcirc\ \bigcirc^*|\right)^* \bigcirc\ \bigcirc\ \bigcirc^* \left(\epsilon + |\,|^*\right)$$

$$\widetilde{L_2} = \bigcirc^*| \left(\bigcirc\ \bigcirc^*|\right)^* |^* \bigcirc \left(\epsilon + |\,|^*\right)$$

Property: $\widetilde{\varphi^c} = (\bigcirc + |)^* \bigcirc\ \bigcirc$

Intersection: $\widetilde{\psi} \cap \widetilde{\varphi^c} = \emptyset$

Thus, using this method we have proved, what we failed to do using regular model checking, that Peterson(N) satisfied mutual exclusion.

Method 2:

Computing with cascade product 1DPDA∘DFA.

1DPDA - single state DPDA with no accepting states

DFA - the DFA looks at the top of the stack of the 1DPDA

Method 2:

Computing with cascade product $\mathrm{lDPDA} \circ \mathrm{DFA}$.

lDPDA

$\sharp(\mid, w) = \sharp(\bigcirc, w)$

$\Delta(0, \bot) = +a \quad \Delta(0, a) = +a \quad \Delta(0, b) = -b$
$\Delta(1, \bot) = +b \quad \Delta(1, a) = -a \quad \Delta(1, b) = +b$

Input Letter Stack symbol Stack operation

Method 2:

Computing with cascade product $\mathrm{lDPDA} \circ \mathrm{DFA}$.

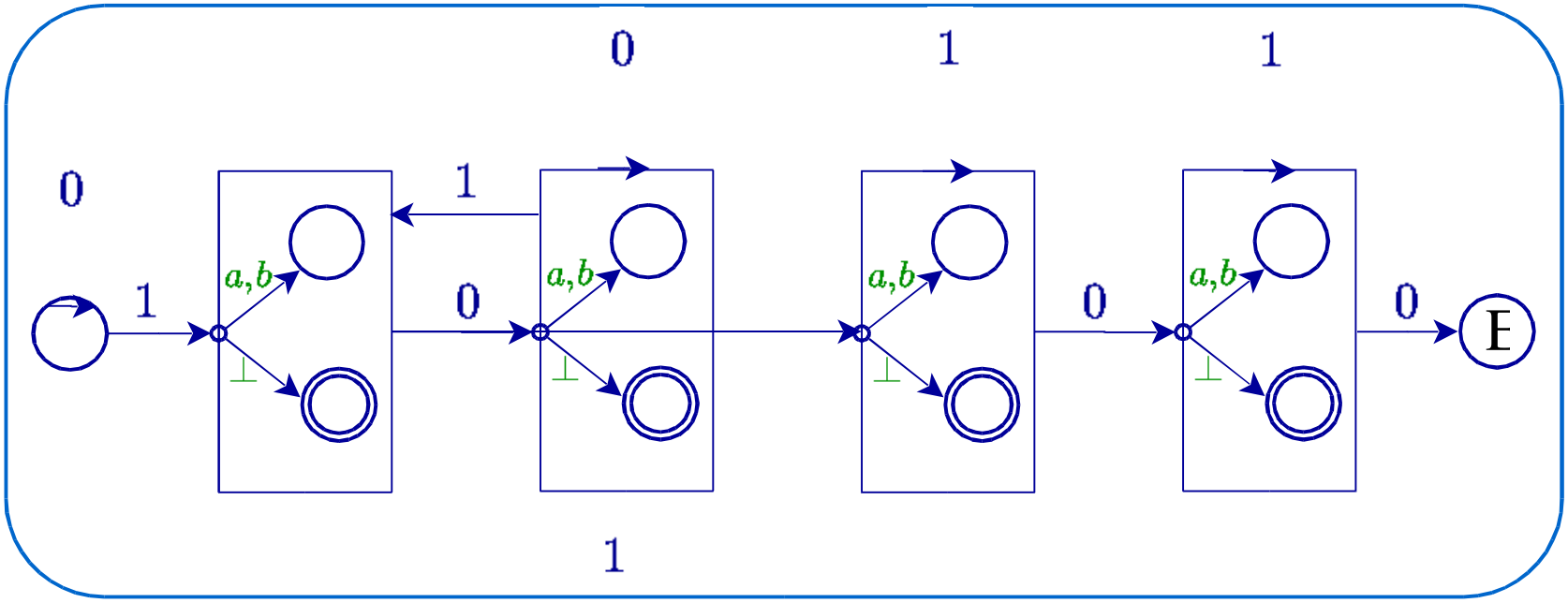
lDPDA

$\sharp(|, w) = \sharp(\bigcirc, w)$

$\Delta(0, \bot) = +a \qquad \Delta(0, a) = +a \qquad \Delta(0, b) = -b$
$\Delta(1, \bot) = +b \qquad \Delta(1, a) = -a \qquad \Delta(1, b) = +b$

DFA $\quad \bigcirc^* \left(| \bigcirc^+ \right)^* \left(|^* + \; ||^+ \bigcirc |^* \right)$

to the right of the first empty partition there is at most one processor



Method 2:

Computing with cascade product $\mathrm{lDPDA} \circ \mathrm{DFA}$.

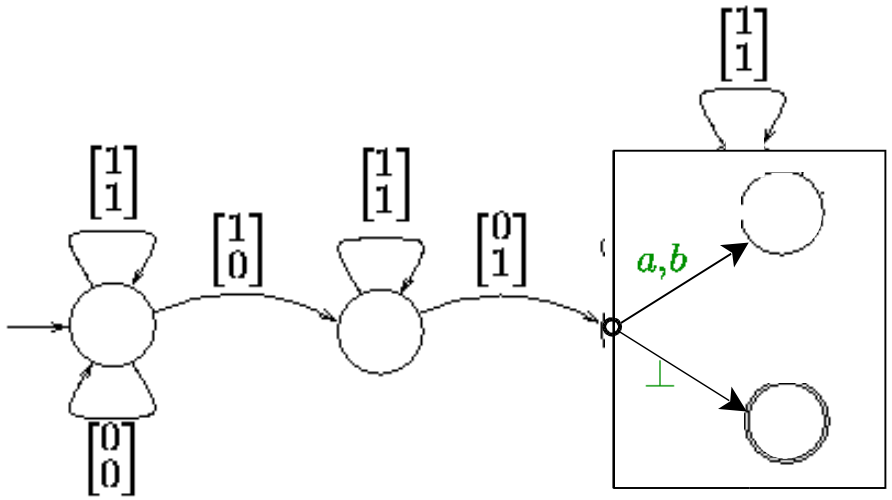
lDPDA

$\sharp(\vert, w) = \sharp(\bigcirc, w)$

$$\Delta(0,\perp) = +a \qquad \Delta(0,a) = +a \qquad \Delta(0,b) = -b$$
$$\Delta(1,\perp) = +b \qquad \Delta(1,a) = -a \qquad \Delta(1,b) = +b$$

DFA $\qquad X\ 1\ Y\ 0\ Z \mapsto X\ 0\ Y\ 1\ Z$

where $X \in \{1,0\}^*$ and $Y, Z \in 1^*$



Beyond Regular

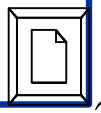
Method 2:

Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$.

The class $\mathcal{L}_{DPDA-M}$

A_1

A_2

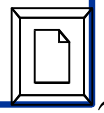


Beyond Regular

Method 2:

Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$.

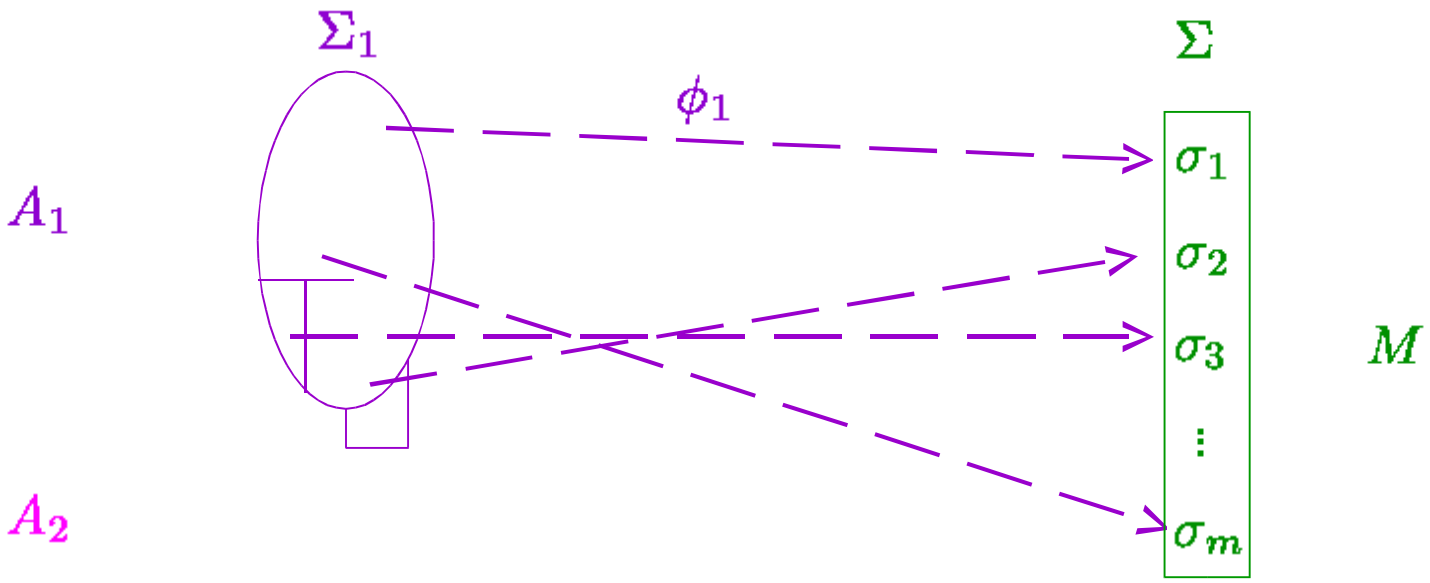
The class $\mathcal{L}_{DPDA-M}$

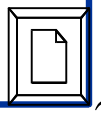


Method 2:

Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$.

The class $\mathcal{L}_{DPDA-M}$



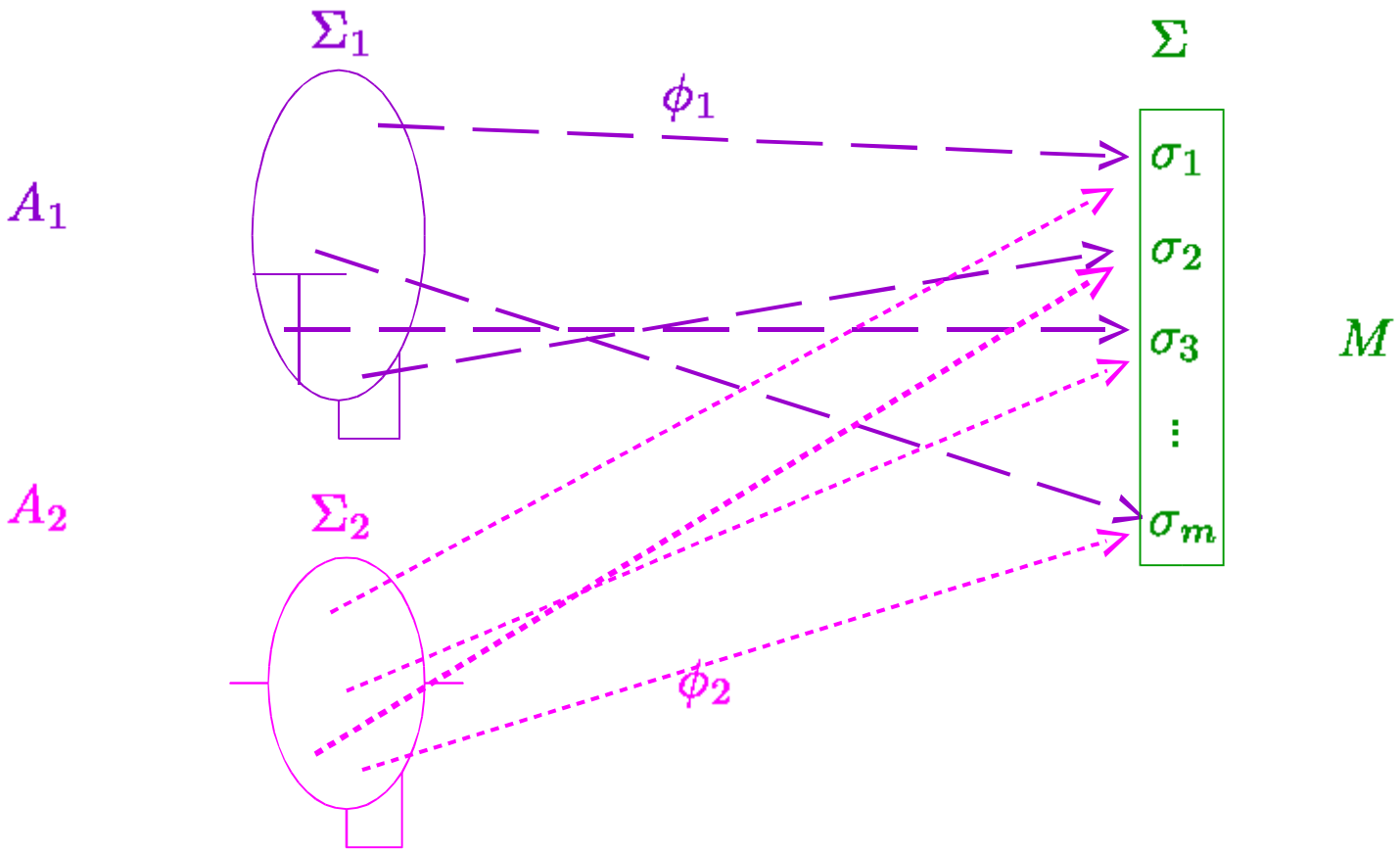


Beyond Regular

Method 2:

Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$.

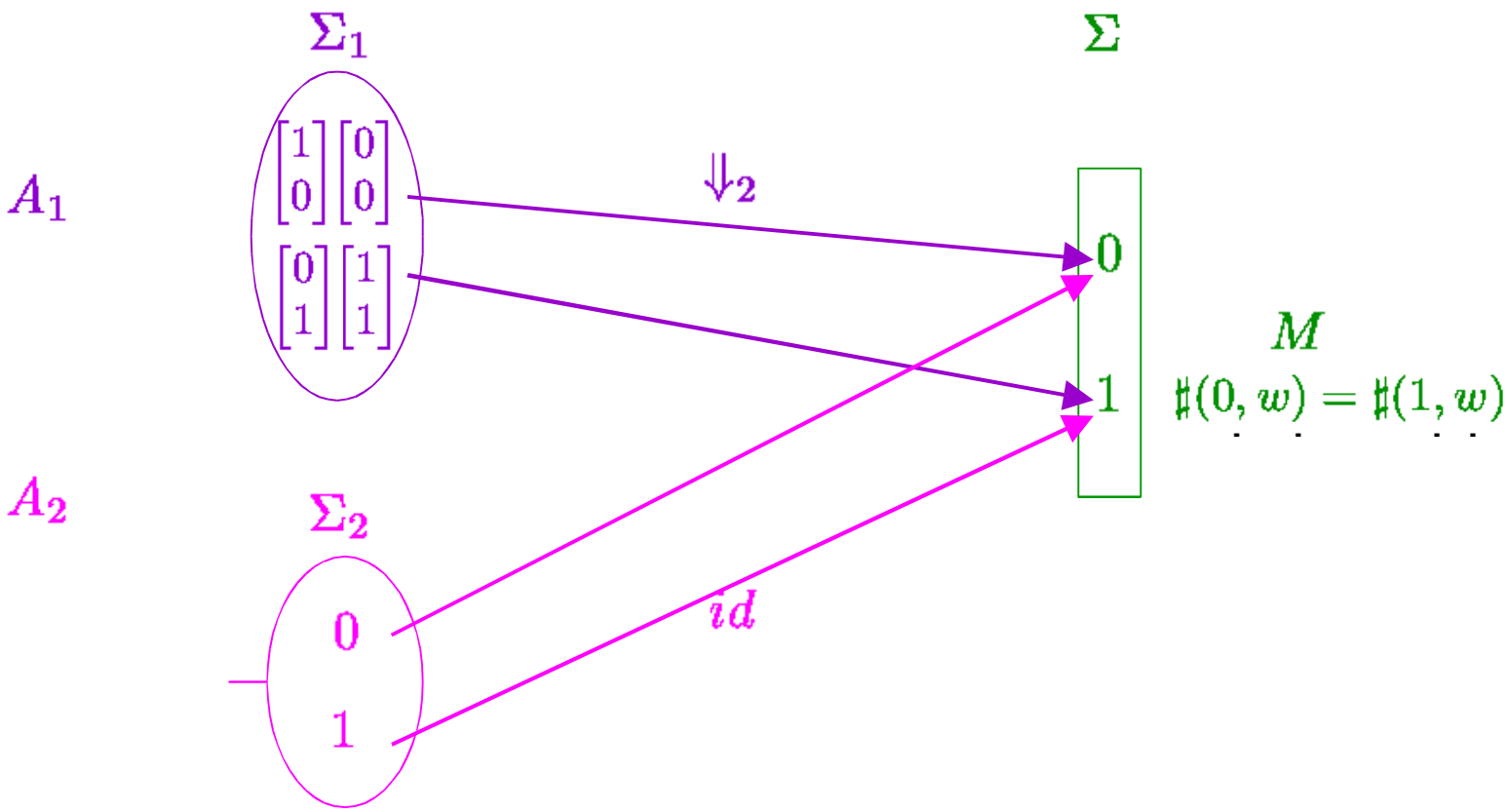
The class $\mathcal{L}_{DPDA-M}$

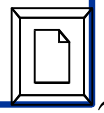


Method 2:

Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$.

The class $\mathcal{L}_{DPDA-M}$





Beyond Regular

Method 2:

Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$.

Computing Projection

Input: $A = M \circ_{\Downarrow_2} R$ over $\Sigma \times \Sigma$

Output: $\widetilde{A} = M \circ_{id} \widetilde{R}$ over Σ

Phase 1: *Reachability Analysis*

- Compute for the initial state the language of stack content

Phase 2: *Annotation*

- Guess stack operations when looking at the unprimed coordinate
- Compute stack difference between primed and unprimed
- Compute stack language of prime for all states

Phase 3: *Projection*

- Project on the unprimed coordinate

Phase 4: *Determinization*

- Compute subset construction

Beyond Regular

Method 2: Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$

The class $\mathcal{L}_{DPDA-M}$

a DPDA A is in the class $\mathcal{L}_{DPDA-M}$ if $A = M \circ_\phi R$ for some DFA R and some substitution ϕ

Method 2: Computing with cascade product $\mathrm{DPDA} \circ \mathrm{DFA}$

The class $\mathcal{L}_{DPDA-M}$

Claim

- The class $\mathcal{L}_{DPDA-M}$ is closed under complementation.
- The class $\mathcal{L}_{DPDA-M}$ is closed under lifting.
- The class $\mathcal{L}_{DPDA-M}$ is closed under intersection with a regular language.
- The Emptiness question is decidable for the class $\mathcal{L}_{DPDA-M}$.
- The Equivalence question is decidable for the class $\mathcal{L}_{DPDA-M}$.

Theorem

There exists a procedure in RE for calculating the projection of languages in $\mathcal{L}_{DPDA-M}$.

Corollary

The class $\mathcal{L}_{DPDA-M}$ is adequate for symbolic model checking (in collaboration with the $\mathcal{L}_{FA}$ class).

Method 3:

Computing with product DPDA × DFA

Method 3:
Computing with product DPDA×DFA

- Start with a product of a DPDA L and a DFA R_0
- Check that L is preserved under R $\qquad L = ((L \times \Sigma^*) \cap R)\Downarrow_2$
- Compute projection only on the DFA part.

Procedure *Forward MC*

$M_0 := L \cap R_0$

For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := L \cap ((R_i \times \Sigma^*) \cap R)\Downarrow_2$

until $M_{i+1} = M_i$

return $M_i \cap \overline{A_\varphi} = \emptyset$

Procedure *Backward MC*

$M_0 := L \cap R_0$

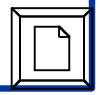
For $i = 0, 1, \ldots$ **repeat**

$M_{i+1} := ((\Sigma^* \times R_i) \cap R)\Downarrow_1 \cap L$

until $M_{i+1} = M_i$

return $M_i \cap A_{Init} = \emptyset$

Beyond Regular



Method 3:
Computing with product DPDA × DFA

Any acceleration technique developed for regular model checking can be applied here as well.

This class is therefore *practically* adequate for symbolic model checking.

Method 3: Computing with product DPDA × DFA

Peterson Example

$$\chi = \left\{ w \in (\bigcirc + |)^* \ : \ \sharp(|, w) = \sharp(\bigcirc, w) > 1 \right\}$$

$$\Theta = \left\{ \bigcirc^i |^i \ : \ i > 1 \right\}$$

ρ_1	:	$u \bigcirc \, \vert v \mapsto u \vert \bigcirc v$	where $u \in \bigcirc^*, v \in (\vert + \bigcirc)^*$
ρ_2	:	$u \vert \bigcirc \, \vert v \mapsto u \vert \, \vert \bigcirc v$	where $u \in (\vert + \bigcirc)^*, v \in \vert^*$
ρ_3	:	$u \bigcirc \bigcirc \, \vert v \mapsto u \bigcirc \vert \bigcirc v$	where $u, v \in (\vert + \bigcirc)^*$
ρ_4	:	$u \bigcirc \mapsto \bigcirc u$	where $u \in (\vert + \bigcirc)^*$
ρ_{id}	:	$u \mapsto u$	where $u \in (\vert + \bigcirc)^*$

Θ is right preserved by ρ thus it is enough to project the DFA part.

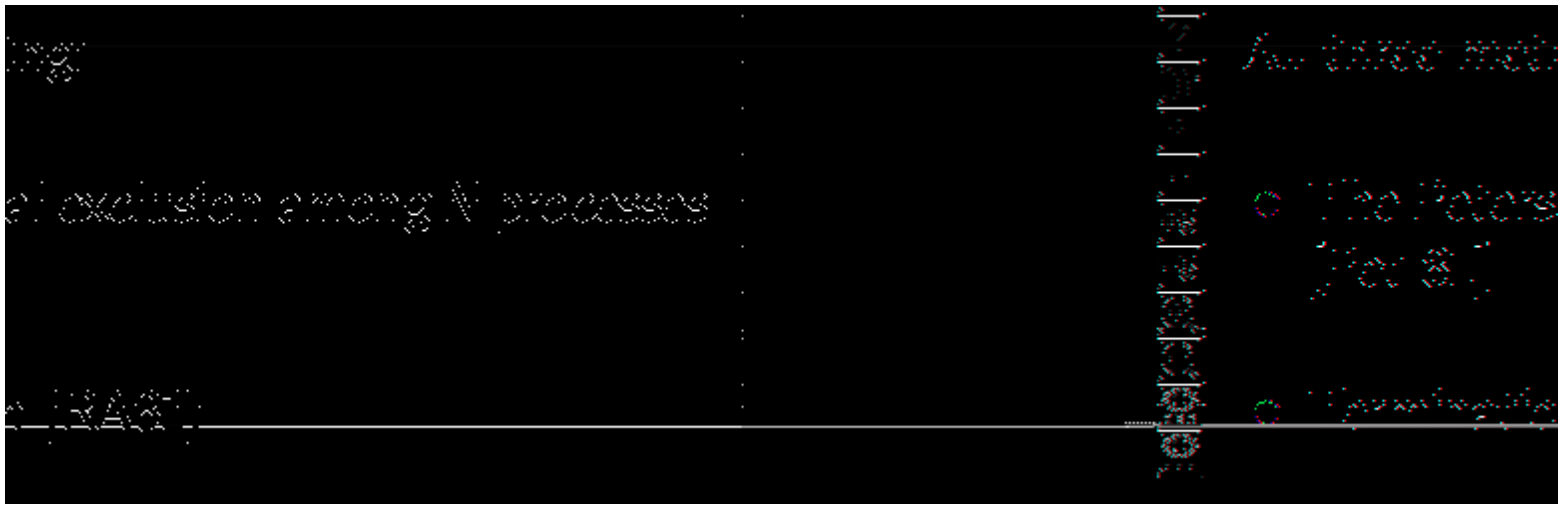
Beyond Regular

Summary

We aimed to find automatic methods for the uniform verification of non-regular parameterized systems.

- Method 1: Computing reachable set by regular means and **then** intersecting with a CFL.
- Method 2: Computing with cascade product $1\mathrm{DPDA} \circ \mathrm{DFA}$.
- Method 3: Computing with product $\mathrm{DPDA} \times \mathrm{DFA}$.

Summary



Related Work

- PushDown Systems

  [BS9596] [WB9292] [BG96] [BEM97] [JES00] [BM96] [BGWW97] [Wal97] [FWW97]

- Systems with Counters

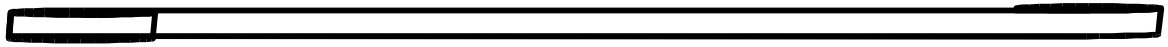
  [Boigelot and Wolper 94][Comon and Jurski 98]

- Combining restricted finite automata and linear constrains on the number of occurrences of symbols

  [Bouajjani and Halbermehl 97]

Future Work

- Verification of the correctness of encoding.
- An equivalent assertional language.
- Use of the $\mathcal{L}_{DPDA-M}$ class in deductive verification.
- Find other classes, stronger than regular languages, that are adequate for SMC.



Beyond Regular