Acceleration in multi pushdown systems

(TACAS’16)

Prakash Saivasan

Joint work with
Mohamed Faouzi Atig
Narayan Kumar K K
Verification of concurrent programs with:

- Programs with multiple threads
- Threads can have recursion
- Finite data domain
- Shared memory
## Formal models

<table>
<thead>
<tr>
<th>Programs</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive</td>
<td>Pushdown Systems</td>
</tr>
<tr>
<td>Concurrent Recursive</td>
<td>Multi-pushdown Systems</td>
</tr>
</tbody>
</table>
Multi-pushdown systems
Multi pushdown Systems

Configuration

Stacks

Stack Alphabets

Transitions

State

Configuration

Stacks

Stack Alphabets

Transitions

Turing powerful
Existing underapproximations

- Bounded Context
- Bounded Phase
- Ordered MPDS
- Bounded Scope
Bounded context

Context is a sequence of operations restricted to a stack

Reachability is NP-Complete

S. Qadeer J. Rehof
Acceleration
Multi pushdown system $\mathcal{M}$

Transitions $\Delta$

Set of configurations $\mathcal{C}$

Set of sequences of transitions $\theta$

Acceleration problem is to compute

$$\{ c' \mid c \xrightarrow{\sigma} c', c \in \mathcal{C}, \sigma \in \theta^* \}$$
INITIAL SET OF CONFIGURATION

FINITE REPRESENTATION
REGULAR/RATIONAL

ACCELERATED SET
Stability

Stability: Representation of initial configuration and the accelerated set are the same.
Bounded context analysis as an acceleration problem
Multi pushdown system $\mathcal{M}$

Transitions $\Delta$

Set of configurations $\mathcal{C}$

Set of sequences of transitions

$$\theta = \bigcup_{i_1, \ldots, i_k \in [1..n]} \Delta_{i_1}^* \Delta_{i_2}^* \cdots \Delta_{i_k}^*$$

We are interested in the following set

$$\{c' \mid c \xrightarrow{\sigma} c, c \in \mathcal{C}, \sigma \in \theta\}$$
Bounded context acceleration

\[ \bigcup_{i_1, \ldots, i_k \in [1..n]} \Delta_{i_1}^* \cdot \Delta_{i_2}^* \cdot \ldots \cdot \Delta_{i_k}^* \]

Initial configuration
Regular

Accelerated set is also regular
Bounded context acceleration

Initial configuration

Rational

Accelerated set is also rational

\[ \bigcup_{i_1, \ldots, i_k \in [1..n]} \Delta^*_i \]
Accelerating loop
Multi pushdown system $\mathcal{M}$

Transitions $\Delta$

Set of configurations $\mathcal{C}$

loop $\theta = \{(q, \text{op}_1, q_1)(q_1, \text{op}_2, q_2) \cdots (q_m, \text{op}_m, q)\}$

$$\{c' \mid c \xrightarrow{\sigma} c', \ c \in \mathcal{C}, \sigma \in \theta^* \}$$
Accelerated set is not regular but rational.

Initial configuration
Regular

\[ \theta = \{(q, op_1, q_1)(q_1, op_2, q_2) \cdots (q_m, op_m, q)\} \]
Accelerating loop on regular set is not regular
Accelerating loop on regular set is rational

We will assume that we are given a set of finite state automata one for each stack recognising the regular set of configurations
Accelerating loop on regular set is rational

- We will first examine the effect of a loop on each stack:

  Stack-1
  1
  1
  1

  Stack-2
  2
  2
  2

  Stack-3
  3
Accelerating loop on regular set is rational

- What is the effect of accelerating the loop repeatedly?

Stack-1

Stack-2

Stack-3
Given a loop, its effect can be summarised as two words for each stack.

The first word is what is removed from the stack at the end of execution of the loop.

The second word is what is appended to the stack at the end of loop execution.

<table>
<thead>
<tr>
<th>Stack</th>
<th>pop word</th>
<th>push word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1↑1↑</td>
</tr>
<tr>
<td>2</td>
<td>ε</td>
<td>2↑</td>
</tr>
<tr>
<td>3</td>
<td>ε</td>
<td>3↑</td>
</tr>
</tbody>
</table>
Accelerating loop on regular set is rational

<table>
<thead>
<tr>
<th>Stack</th>
<th>pop</th>
<th>push</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>u1</td>
<td>v1</td>
</tr>
<tr>
<td>2</td>
<td>u2</td>
<td>v2</td>
</tr>
<tr>
<td>3</td>
<td>u3</td>
<td>v3</td>
</tr>
</tbody>
</table>

Accelerating loop once amounts to removing pop word and adding push word to the stack, what about accelerating multiple times?

CANNOT BE ACCELERATED MORE THAN ONCE

Acceleration is possible only if pop word is prefix of push word or push word is prefix of pop word.
Accelerating loop on regular set is rational

<table>
<thead>
<tr>
<th>Stack</th>
<th>pop</th>
<th>push</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( u_1 )</td>
<td>( v_1 )</td>
</tr>
<tr>
<td>2</td>
<td>( u_2 )</td>
<td>( v_2 )</td>
</tr>
<tr>
<td>3</td>
<td>( u_3 )</td>
<td>( v_3 )</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\mathbf{u}_i &< \text{pre} \mathbf{v}_i \\
\mathbf{v}_i & = \mathbf{u}_i \mathbf{y}_i, \mathbf{x}_i = \epsilon \\
\mathbf{v}_i &< \text{pre} \mathbf{u}_i \\
\mathbf{u}_i & = \mathbf{v}_i \mathbf{x}_i, \mathbf{y}_i = \epsilon
\end{align*}
\]
Accelerating loop $j+1$ times

<table>
<thead>
<tr>
<th>Stack</th>
<th>pop</th>
<th>push</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$u_1 x_1^j$</td>
<td>$v_1 y_1^j$</td>
</tr>
<tr>
<td>2</td>
<td>$u_2 x_2^j$</td>
<td>$v_2 y_2^j$</td>
</tr>
<tr>
<td>3</td>
<td>$u_3 x_3^j$</td>
<td>$v_3 y_3^j$</td>
</tr>
</tbody>
</table>
Accelerating loop on regular set is rational

We construct an $2n$ tape rational automata.
Accelerating loop on regular set is rational

\[
\begin{array}{cccc}
  u_1 & v_1 & u_2 & v_2 \\
  \cdots & \cdots & \cdots & \cdots \\
  x_1 & y_1 & x_2 & y_2 \\
  x_1 & y_1 & x_2 & y_2 \\
  w_1 & w_1 & w_2 & w_2 \\
  \times & \times & \times & \times \\
  B_1 & B_2 & \cdots & B_n
\end{array}
\]
Accelerating loop on rational set is not rational
Constrained simple regular expression
Constrained Simple Regular Expression (1-dim)

\[ w_1 \quad w_2 \quad \cdots \quad \cdots \quad w_n \]

\[ x_1 \quad x_2 \quad \cdots \quad x_n \]

PRESBURGER FORMULA \( \psi \)

CSRE is given by sequence of words and a Presburger formula with one free variable per every word.
Acceptance in CSRE

\[ i_1, i_2, \ldots, i_n \models \Psi(x_1, \ldots, x_n) \]
Constrained Simple Regular Expression (m-dim)

\[
\begin{array}{cccc}
  w_1^1 & w_2^1 & \cdots & \cdots & w_n^1 \\
  w_1^2 & w_2^2 & \cdots & \cdots & \quad \\
  \vdots & \vdots & \ddots & \ddots & \vdots \\
  w_1^m & w_2^m & \cdots & \cdots & w_n^m \\
\end{array}
\]

\[
\begin{array}{cccc}
  x_1^1 & x_1^2 & \cdots & \cdots & x_1^n \\
  x_2^1 & x_2^2 & \cdots & \cdots & x_n^1 \\
  \vdots & \vdots & \ddots & \ddots & \vdots \\
\end{array}
\]

PRESBURGER FORMULA \[\Psi\]
Acceptance in m-dim CSRE

\[ i_1^1, i_1^2, \ldots, i_1^n, \ldots, i_m^1, \ldots, i_m^n \models \Psi \]
\[ \theta = \{(q, op_1, q_1)(q_1, op_2, q_2) \cdots (q_m, op_m, q)\} \]
Some properties of CSRE

- CSRE are closed under intersection, union and concatenation.
- Emptiness, membership and inclusion problems are decidable.
- CSRE is closed under left quotient.
Accelerating loop on CSRE
Accelerating loop $j+1$ times

<table>
<thead>
<tr>
<th>Stack</th>
<th>pop word</th>
<th>push word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$u_1 x_1^j$</td>
<td>$v_1 y_1^j$</td>
</tr>
<tr>
<td>2</td>
<td>$u_2 x_2^j$</td>
<td>$v_2 y_2^j$</td>
</tr>
<tr>
<td>3</td>
<td>$u_3 x_3^j$</td>
<td>$v_3 y_3^j$</td>
</tr>
</tbody>
</table>
We first left quotient $u_1, \ldots, u_n$

We left quotient $x^j_1, \ldots, x^j_n$

We concatenate $y^j_1, \ldots, y^j_n$

We concatenate $v_1, \ldots, v_n$

Presburger formula is used to ensure concatenation and left quotient is done same number of times
Context switch set
Context switch set

- Loops are weak, cannot capture bounded context switch

- We will introduce notion of context switch set

\[ \Lambda = \tau_1^* \sigma_1 \tau_2^* \sigma_2 \cdots \tau_n^* \sigma_n \]

- \( \tau_i \) Subset of transitions operating on a stack
- \( \sigma_i \) Single transition
\[ \Lambda = \tau_1 \sigma_1 \tau_2 \sigma_2 \cdots \tau_n \sigma_n \]
Constrained Rational Automata
Constrained Rational Automata (Parikh automata)

Multi tape automata

\[
\begin{array}{cccc}
q_1 & q_4 & q_2 \\
q_3 & q_m & \\
\tau_1 & \tau_2 & \tau_3 & \cdots & \tau_m \\
\end{array}
\]

\[
\begin{array}{ccc}
\Sigma_1 & \Sigma_2 & \Sigma_3 \\
\Sigma_4 & \Sigma_n & \\
\end{array}
\]

Presburger formula

\[\Phi\]
Some properties of Constrained Rational Automata (CRA)

- CRA are closed under concatenation, union but not under intersection.
- Emptiness and membership problems are decidable.
Create a pushdown systems, one for each stack.
- Each of the pushdown system simulates moves of stack-\(i\).
- It further has jump transitions corresponding to \(\sigma_1, \cdots, \sigma_n\).
- It outputs number of times a jump transition was made.
- The following language for any pushdown is rational:

\[
\{(v, w, \#^j) \mid (q, v) \overset{\#^j}{\rightarrow} (q, v')\}
\]

- We can use Presburger formula to ensure that the number of jumps of each PDS match.
THANK YOU