Collapsible Pushdown Graphs of Level 2 are Tree-Automatic

Alexander Kartzow
Technische Universität Darmstadt
Department of Mathematics
kartzow@mathematik.tu-darmstadt.de

Supported by DFG Grant OT147/5-1

Collapsible Pushdown Systems (of order 2)

Theorem (Hague, Murawski, Ong, Serre)

For collapsible pushdown graphs:

- *modal μ-calculus model checking: decidable*
- *MSO model checking: undecidable*

What about FO on CPG? Today: For level 2: decidable

Level 2 Collapsible Pushdown Graphs (CPG)

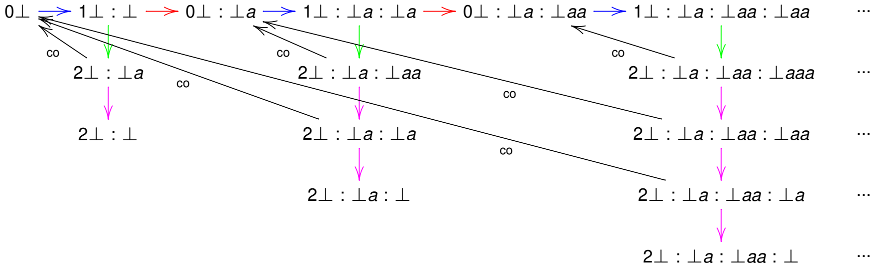
TECHNISCHE UNIVERSITÄT DARMSTADT

- Collapsible pushdown system with stack of stacks
- Operations: $\text{push}_\sigma, \text{clone}, \text{pop}_1, \text{pop}_2, \text{collapse}$
- Configurations: (q, s) – q a state, s a stack
  Edges: $(q, s) \xrightarrow{op} (q', s')$
  for transitions $(q, \text{top}_1(s)) \longrightarrow (q', op)$ with $op(s) = s'$
- CPG: Graph of *reachable* configurations with labeled transitions

Example

TECHNISCHE UNIVERSITÄT DARMSTADT

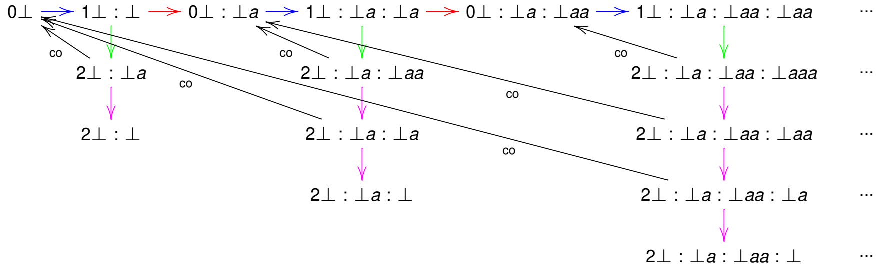
$(0, *, 1, \text{clone}), (1, *, 0, \text{push}_a), (1, *, 2, \text{push}_a), (2, a, 2, \text{pop}_1), (2, a, 0, \text{collapse})$



Example

TECHNISCHE UNIVERSITÄT DARMSTADT

$(0, *, 1, \text{clone}), (1, *, 0, \text{push}_a), (1, *, 2, \text{push}_a), (2, a, 2, \text{pop}_1), (2, a, 0, \text{collapse})$



$\rightarrow$: access to “same column”

$\rightarrow$: access to “same diagonal”

$\Rightarrow$ grid is MSO-definable $\Rightarrow$ MSO undecidable on CPG

Tree-automaticity

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition (of tree-automaton)

Tree-automaton reads finite binary tree
Labels nodes from the root down to the leaves
according to $\Delta \subseteq Q \times \Sigma \times Q \times Q$
Accepts, if all leaves are labeled by final states

Definition

A structure $\mathbb{S} := (S, E_1, E_2, \dots, E_n)$ is tree-automatic iff there are tree-automata $A_S, A_{E_1}, \dots, A_{E_n}$ and a bijection $f : S \rightarrow L(A_S)$ such that
$(s_1, s_2) \in E_i$ iff A_{E_i} accepts $f(s_1) \otimes f(s_2)$

Theorem

FO model checking on every tree-automatic structure is decidable.

CPG are tree-automatic

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Theorem

CPG are tree-automatic (even when extended by regular reachability predicates).

Proof by

- Encoding stacks $\longrightarrow$ trees
- stack-operations $\longrightarrow$ easy tree-operations.
- reachable configurations $\longrightarrow$ accepted trees.

Corollary (To, Libkin LPAR 2008)

Recurrent Reachability on CPG is decidable.

Idea of Encoding

TECHNISCHE UNIVERSITÄT DARMSTADT

$\bot \ \sigma_1 \qquad w_1$

$\bot \ \sigma_1 \qquad w_2$

$\vdots$

$\bot \ \sigma_1 \qquad w_3$

$\bot \ \sigma_2 \qquad w_4$

$\vdots$

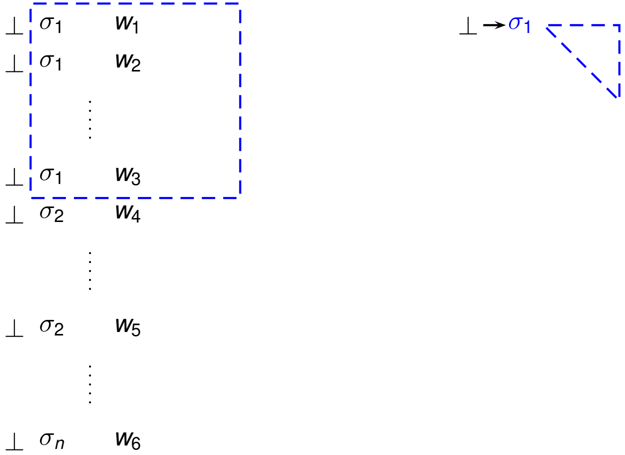
$\bot \ \sigma_2 \qquad w_5$

$\vdots$

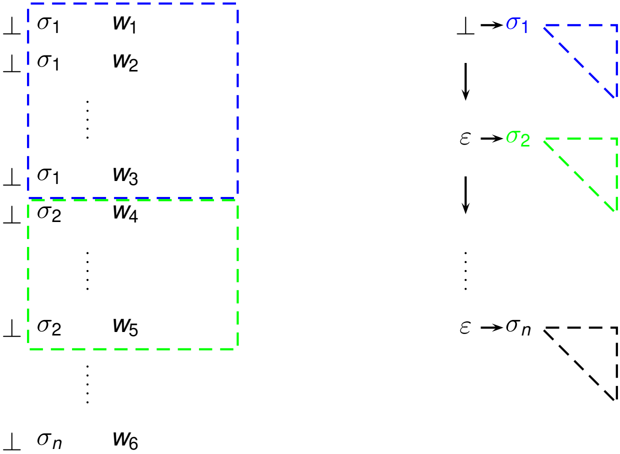
$\bot \ \sigma_n \qquad w_6$

Idea of Encoding

TECHNISCHE UNIVERSITÄT DARMSTADT



Idea of Encoding



Encoding Example

TECHNISCHE UNIVERSITÄT DARMSTADT

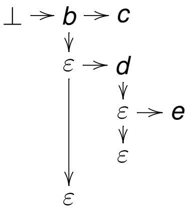
$\bot \quad b \quad c$

$\bot \quad b \quad d$

$\bot \quad b \quad d \quad e$

$\bot \quad b \quad d$

$\bot \quad b$

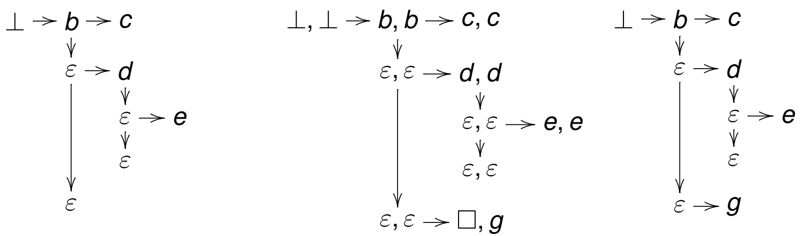


Encoding Example

$\bot$	b	c	
$\bot$	b	d	
$\bot$	b	d	e
$\bot$	b	d	
$\bot$	b		

push_g

$\bot$	b	c	
$\bot$	b	d	
$\bot$	b	d	e
$\bot$	b	d	
$\bot$	b	g	

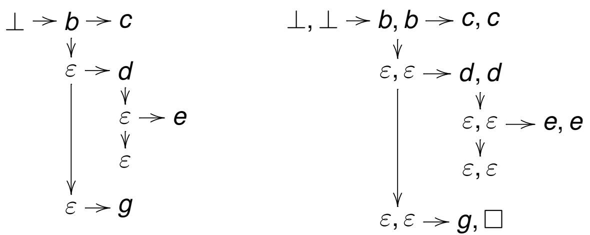


Encoding Example

$\bot$	b	c	
$\bot$	b	d	
$\bot$	b	d	e
$\bot$	b	d	
$\bot$	b	g	

pop_1

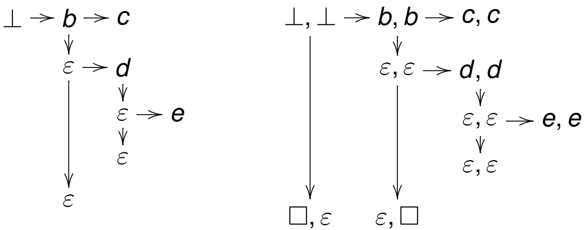
$\bot$	b	c	
$\bot$	b	d	
$\bot$	b	d	e
$\bot$	b	d	
$\bot$	b		



Encoding Example

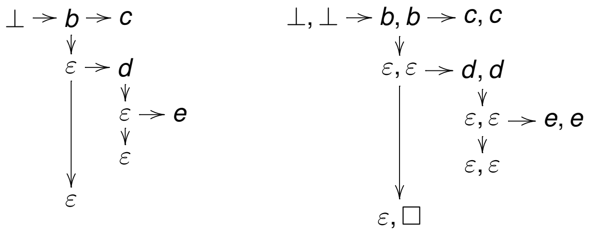
TECHNISCHE UNIVERSITÄT DARMSTADT

$\bot$	b	c				$\bot$	b	c		
$\bot$	b	d				$\bot$	b	d		
$\bot$	b	d	e		pop_1	$\bot$	b	d	e	
$\bot$	b	d				$\bot$	b	d		
$\bot$	b					$\bot$				



Encoding Example

$\bot$	b	c			$\bot$	b	c	
$\bot$	b	d			$\bot$	b	d	
$\bot$	b	d	e	pop_2	$\bot$	b	d	e
$\bot$	b	d			$\bot$	b	d	
$\bot$	b							



Encoding Example

TECHNISCHE UNIVERSITÄT DARMSTADT

$\bot \quad b \quad c$

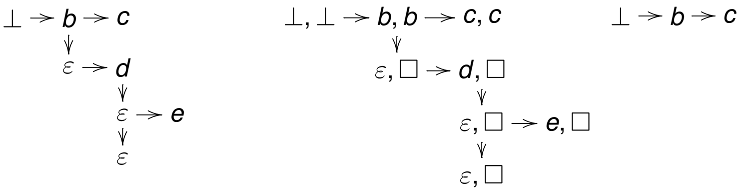
$\bot \quad b \quad d$

$\bot \quad b \quad d \quad e$

$\bot \quad b \quad d$

collapse

$\bot \quad b \quad c$



Detecting Reachable Configurations

- Nodes in a CPG: *reachable* configurations
- Need: A accepts T if $(q, s) = \text{Decode}(T)$ reachable in CPG
- Idea:
  1. Milestones: Necesarily passed substacks on a run to (q, s)
  2. Identify milestons of s with nodes of T
  3. Guess at $t \in T$ the state of the corresponding milestone
  4. Verify this guess. Problem: loops with large stacks

Milestones

TECHNISCHE UNIVERSITÄT DARMSTADT

Milestones of s: Necesarily passed substacks on every run to (q, s)

$q_0, \bot \dashrightarrow \qquad q_4, \bot ab$

$\bot ac$

Milestones

TECHNISCHE UNIVERSITÄT DARMSTADT

Milestones of s: Necesarily passed substacks on every run to (q, s)

$$q_0, \bot \dashrightarrow q_1, \bot a \dashrightarrow \qquad q_4, \bot ab$$

$$\bot ac$$

Milestones

TECHNISCHE UNIVERSITÄT DARMSTADT

Milestones of s: Necesarily passed substacks on every run to (q, s)

$$q_0, \bot \dashrightarrow q_1, \bot a \dashrightarrow q_2, \bot ab \dashrightarrow \qquad q_4, \bot ab$$

$$\bot ac$$

Milestones

Milestones of s: Necesarily passed substacks on every run to (q, s)

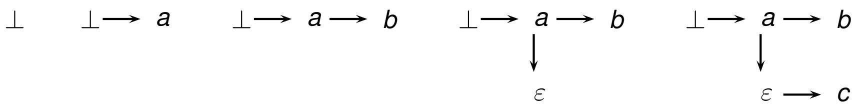
$$q_0, \bot \dashrightarrow q_1, \bot a \dashrightarrow q_2, \bot ab \dashrightarrow \begin{matrix} q_3, \bot ab \\ \bot a \end{matrix} \dashrightarrow \begin{matrix} q_4, \bot ab \\ \bot ac \end{matrix}$$

Milestones

Milestones of s: Necesarily passed substacks on every run to (q, s)

$$\begin{array}{lllll} q_0, \bot \dashrightarrow & q_1, \bot a \dashrightarrow & q_2, \bot ab \dashrightarrow & q_3, \bot ab \dashrightarrow & q_4, \bot ab \\ & & & \quad \bot a & \quad \bot ac \end{array}$$

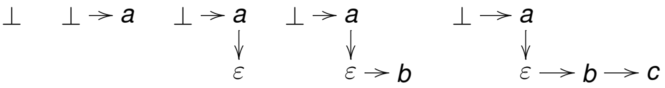
Milestones encoded in the encoding of s:



Example Reachable Configurations

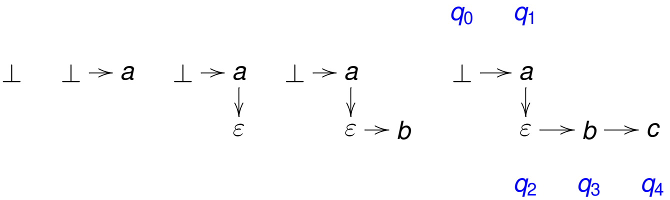
TECHNISCHE UNIVERSITÄT DARMSTADT

$$
\begin{array}{ccccccccc}
q_0, \bot & \longrightarrow & \bot a & \longrightarrow & \bot a & \longrightarrow & \bot a & \longrightarrow & q_4, \bot a \\
 & & & & \bot a & & \bot ab & & \bot abc
\end{array}
$$



Example Reachable Configurations

$$
\begin{array}{ccccccccc}
q_0, \bot & \longrightarrow & q_1, \bot a & \longrightarrow & q_2, \bot a & \longrightarrow & q_3, \bot a & \longrightarrow & q_4, \bot a \\
 & & & & \bot a & & \bot ab & & \bot abc
\end{array}
$$



The Loop Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition

A run r from (q_1, s) to (q_2, s) is a *loop of s* if it does not pass $\text{pop}_2(s)$.

Task for the automaton: Determine $\text{Loops}(s) := \{(q_1, q_2) \text{ s. t. } \exists \text{ loop } q_1, s \text{ to } q_2, s\}$?

The Loop Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition

A run r from (q_1, s) to (q_2, s) is a *loop of s* if it does not pass $\text{pop}_2(s)$.

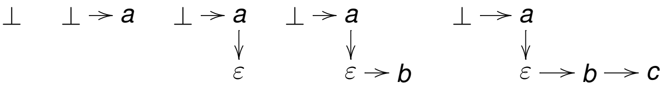
Task for the automaton: Determine $\text{Loops}(s) := \{(q_1, q_2) \text{ s. t. } \exists \text{ loop } q_1, s \text{ to } q_2, s\}$?
Solution:

Lemma (Loop-Lemma)

There is a finite automaton that calculates $\text{Loops}(s)$ *when reading the topmost word of s.*

Example Reachable Configurations II

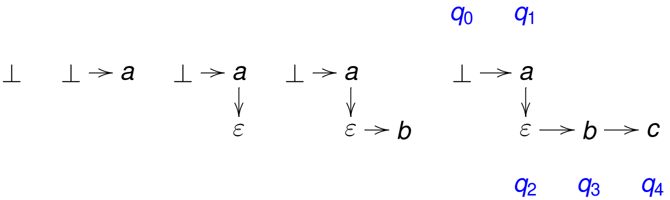
$$q_0, \bot \xrightarrow[\text{push}_a]{\text{Loop+}} \bot a \xrightarrow[\text{clone}]{\text{Loop+}} \begin{array}{l}\bot a\\ \bot a\end{array} \xrightarrow[\text{push}_b]{\text{Loop+}} \begin{array}{l}\bot a\\ \bot ab\end{array} \xrightarrow[\text{push}_c]{\text{Loop+}} q_4, \begin{array}{l}\bot a\\ \bot abc\end{array}$$



Example Reachable Configurations II

TECHNISCHE UNIVERSITÄT DARMSTADT

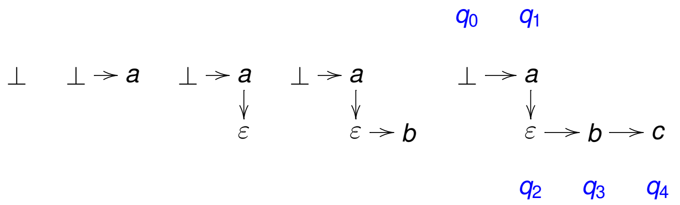
$$q_0, \bot \xrightarrow[\text{push}_a]{\text{Loop+}} \bot a \xrightarrow[\text{clone}]{\text{Loop+}} \begin{matrix}\bot a\\ \bot a\end{matrix} \xrightarrow[\text{push}_b]{\text{Loop+}} \begin{matrix}\bot a\\ \bot ab\end{matrix} \xrightarrow[\text{push}_c]{\text{Loop+}} q_4, \begin{matrix}\bot a\\ \bot abc\end{matrix}$$



Labelling valid if

Example Reachable Configurations II

$$q_0, \bot \xrightarrow[\text{push}_a]{\text{Loop+}} q_1, \bot a \xrightarrow[\text{clone}]{\text{Loop+}} \begin{matrix}\bot a \\ \bot a\end{matrix} \xrightarrow[\text{push}_b]{\text{Loop+}} \begin{matrix}\bot a \\ \bot ab\end{matrix} \xrightarrow[\text{push}_c]{\text{Loop+}} q_4, \begin{matrix}\bot a \\ \bot abc\end{matrix}$$

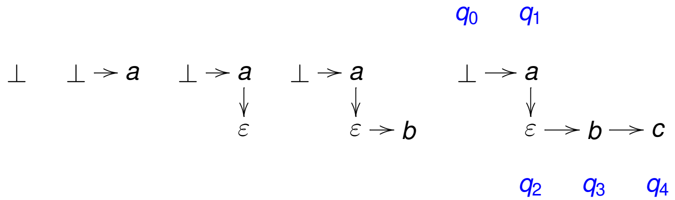


$$\exists q_0' \ \exists \text{loop:} \quad q_0, \bot \to q_0', \bot \text{ and } (q_0', \bot, q_1, \text{push}_a) \in \Delta$$

Labelling valid if

Example Reachable Configurations II

$$q_0, \bot \xrightarrow[\text{push}_a]{\text{Loop+}} q_1, \bot a \xrightarrow[\text{clone}]{\text{Loop+}} q_2, \begin{matrix}\bot a\\ \bot a\end{matrix} \xrightarrow[\text{push}_b]{\text{Loop+}} \begin{matrix}\bot a\\ \bot ab\end{matrix} \xrightarrow[\text{push}_c]{\text{Loop+}} q_4, \begin{matrix}\bot a\\ \bot abc\end{matrix}$$



Labelling valid if

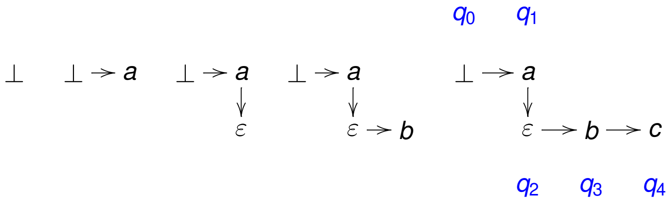
$$\exists q_0' \, \exists \text{loop:} \quad q_0, \bot \to q_0', \bot \text{ and } (q_0', \bot, q_1, \text{push}_a) \in \Delta$$

$$\exists q_1' \, \exists \text{loop:} \quad q_1, \bot a \to q_1', \bot a \text{ and } (q_1', a, q_2, \text{clone}) \in \Delta$$

Example Reachable Configurations II

TECHNISCHE UNIVERSITÄT DARMSTADT

$$q_0, \bot \xrightarrow[\text{push}_a]{\text{Loop+}} q_1, \bot a \xrightarrow[\text{clone}]{\text{Loop+}} q_2, \begin{matrix}\bot a\\ \bot a\end{matrix} \xrightarrow[\text{push}_b]{\text{Loop+}} q_3, \begin{matrix}\bot a\\ \bot ab\end{matrix} \xrightarrow[\text{push}_c]{\text{Loop+}} q_4, \begin{matrix}\bot a\\ \bot abc\end{matrix}$$

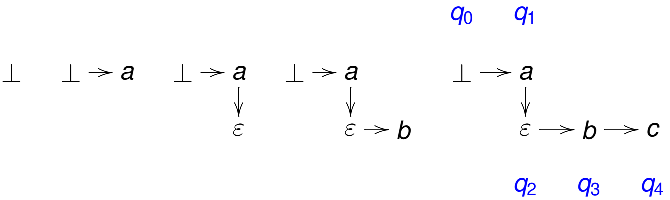


Labelling valid if

$$\exists q_0' \; \exists \text{loop:} \quad q_0, \bot \to q_0', \bot \text{ and } (q_0', \bot, q_1, \text{push}_a) \in \Delta$$
$$\exists q_1' \; \exists \text{loop:} \quad q_1, \bot a \to q_1', \bot a \text{ and } (q_1', a, q_2, \text{clone}) \in \Delta$$
$$\exists q_2' \; \exists \text{loop:} \quad q_2, \bot a \to q_2', \bot a \text{ and } (q_2', a, q_3, \text{push}_b) \in \Delta$$

Example Reachable Configurations II

$$q_0, \bot \xrightarrow[\text{push}_a]{\text{Loop+}} q_1, \bot a \xrightarrow[\text{clone}]{\text{Loop+}} q_2, \begin{array}{l}\bot a\\ \bot a\end{array} \xrightarrow[\text{push}_b]{\text{Loop+}} q_3, \begin{array}{l}\bot a\\ \bot ab\end{array} \xrightarrow[\text{push}_c]{\text{Loop+}} q_4, \begin{array}{l}\bot a\\ \bot abc\end{array}$$



Labelling valid if

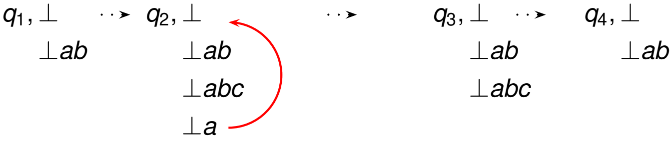
$$\exists q_0' \ \exists \text{loop:} \quad q_0, \bot \rightarrow q_0', \bot \text{ and } (q_0', \bot, q_1, \text{push}_a) \in \Delta$$
$$\exists q_1' \ \exists \text{loop:} \quad q_1, \bot a \rightarrow q_1', \bot a \text{ and } (q_1', a, q_2, \text{clone}) \in \Delta$$
$$\exists q_2' \ \exists \text{loop:} \quad q_2, \bot a \rightarrow q_2', \bot a \text{ and } (q_2', a, q_3, \text{push}_b) \in \Delta$$
$$\exists q_3' \ \exists \text{loop:} \quad q_3, \bot ab \rightarrow q_3', \bot ab \text{ and } (q_3', \bot, q_4, \text{push}_c) \in \Delta$$

Proof of Loop Lemma (1)

Definition

A run from (q_1, s) to $(q_2, \text{pop}_2(s))$ is a *return* if no link in s points to $\text{pop}_2(s)$.

Returns occur as subruns of loops:



Links of topmost word *all* point below s

Proof of Loop Lemma (2)

Stack s with topmost word w A loop of s consists of sequences of

1. pop_1+ Loop of $\text{pop}_1(s)$ +push_a: depends on Loops($\text{pop}_1(s)$)
2. run reaches s' with topmost word $\text{pop}_1(w) \Rightarrow$ run continues with return: depends on Returns($\text{pop}_1(s)$)

2. holds because returns and loops only depend on topmost word

Lemma (Return-Lemma)

Returns(s) *only depend on* Returns($pop_1(s)$) *and the topmost symbol of s.*

Proof.

Same argument as in 2. □

Reachability Predicates

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition

Reachxy holds if there is a path from x to y.

Lemma

Reach *is a tree-automatic relation for all* 2*-CPG.*

Proof.

x substack of $y \Rightarrow$ Use ideas of detecting valid configurations
y substack of $x \Rightarrow$ More and technical variants of loops and returns, similar labelling algorithm. □

Regular Reachability Predicates

Definition

$\text{Reach}_R xy$ holds if there is a path from x to y such that its labels form a word from R.

Lemma

Reach_R *is a tree-automatic relation for all* 2*-CPG and regular sets* R*.*

Proof.

CPS are closed under product with finite automata. □

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

Known results for CPG:

- modal μ-calculus model checking decidable
- MSO model checking undecidable

New result:

- Decidable FO(Reg) model checking on 2-CPG
- Decidable recurrent reachability problem

Proof:

- Tree-automaticity of 2-CPG
- Still tree-automatic with regular reachability predicates

Still open:

- FO model checking on arbitrary CPG?
- Are 3-CPG tree-automatic?