

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

Theorem (Hague, Murawski, Ong, Serre)

For collapsible pushdown graphs:

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

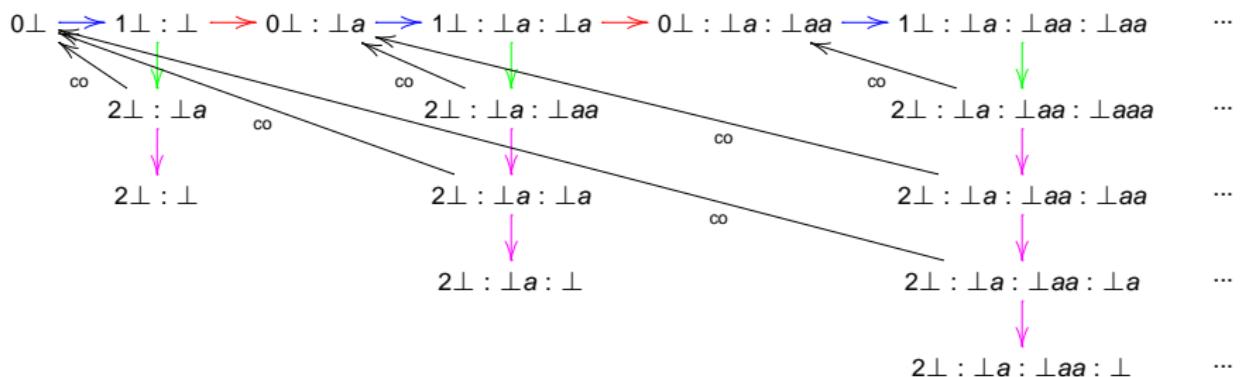
What about FO on CPG? Today: For level 2: decidable

Level 2 Collapsible Pushdown Graphs (CPG)

- ▶ Collapsible pushdown system with stack of stacks
- ▶ Operations: push_σ , clone, pop_1 , pop_2 , collapse
- ▶ Configurations: (q, s) – q a state, s a stack
Edges: $(q, s) \xrightarrow{\text{op}} (q', s')$
for transitions $(q, \text{top}_1(s)) \rightarrow (q', \text{op})$ with $\text{op}(s) = s'$
- ▶ CPG: Graph of *reachable* configurations with labeled transitions

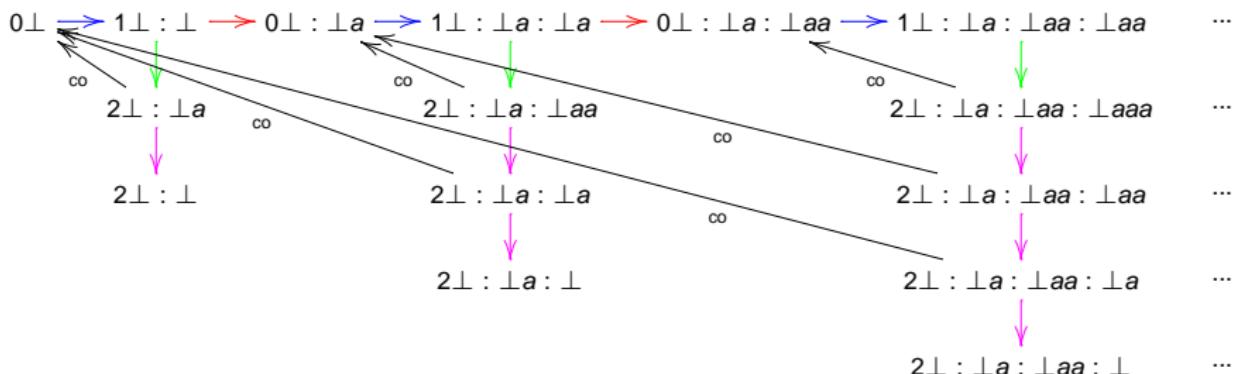
Example

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



Example

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



→: access to “same column”

→: access to “same diagonal”

⇒ grid is MSO-definable ⇒ MSO undecidable on CPG

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.

Theorem

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

Proof by

- ▶ Encoding stacks → trees
- ▶ stack-operations → easy tree-operations.
- ▶ reachable configurations → accepted trees.

Corollary (To, Libkin LPAR 2008)

Recurrent Reachability on CPG is decidable.

Idea of Encoding



TECHNISCHE
UNIVERSITÄT
DARMSTADT

$\perp \sigma_1 w_1$

$\perp \sigma_1 w_2$

⋮

$\perp \sigma_1 w_3$

$\perp \sigma_2 w_4$

⋮

$\perp \sigma_2 w_5$

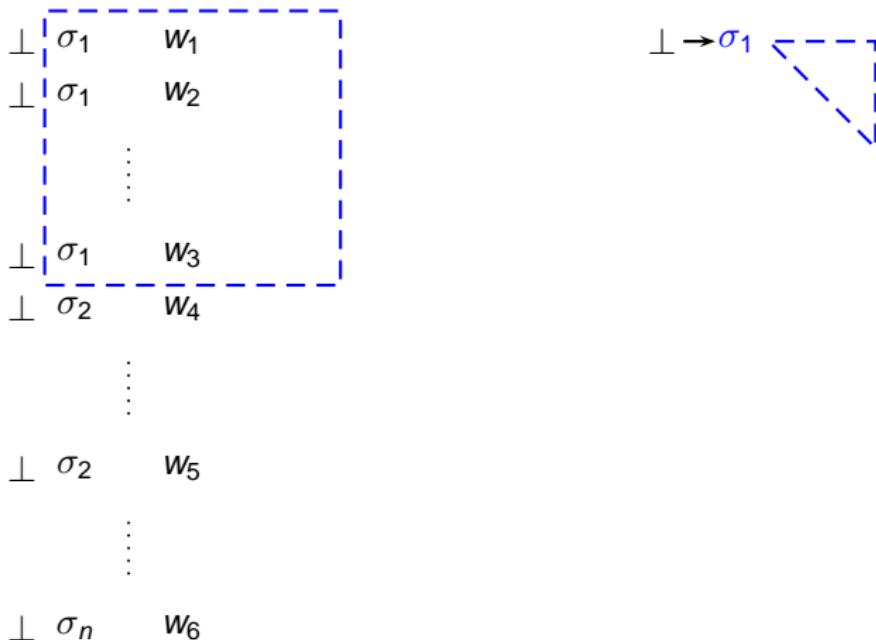
⋮

$\perp \sigma_n w_6$

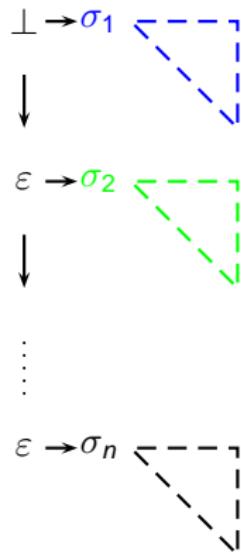
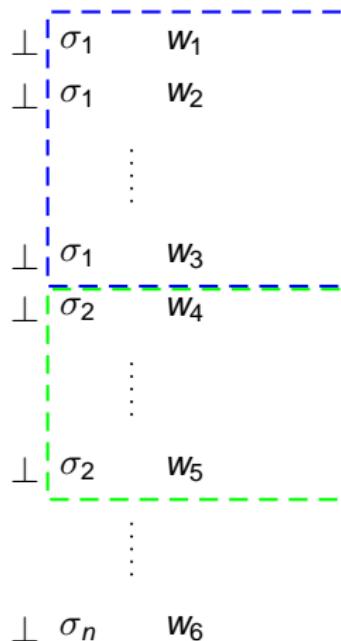
Idea of Encoding



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Idea of Encoding

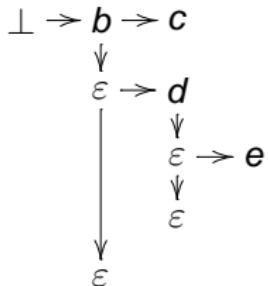


Encoding Example



TECHNISCHE
UNIVERSITÄT
DARMSTADT

$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$
 $\perp \quad b$



Encoding Example



$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$
 $\perp \quad b$

push_g

$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$
 $\perp \quad b \quad g$

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε

$\perp, \perp \rightarrow b, b \rightarrow c, c$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow d, d$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow e, e$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow \square, g$

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε

Encoding Example



$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$
 $\perp \quad b \quad g$

pop_1

$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$
 $\perp \quad b$

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε
 \downarrow
 $\varepsilon \rightarrow g$

$\perp, \perp \rightarrow b, b \rightarrow c, c$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow d, d$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow e, e$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow g, \square$

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε

Encoding Example



$$\begin{array}{c} \perp \quad b \quad c \\ \perp \quad b \quad d \\ \perp \quad b \quad d \quad e \\ \perp \quad b \quad d \\ \perp \quad b \end{array}$$

pop_1

$$\begin{array}{c} \perp \quad b \quad c \\ \perp \quad b \quad d \\ \perp \quad b \quad d \quad e \\ \perp \quad b \quad d \\ \perp \end{array}$$

$$\begin{array}{c} \perp \rightarrow b \rightarrow c \\ \downarrow \\ \varepsilon \rightarrow d \\ \downarrow \\ \varepsilon \rightarrow e \\ \downarrow \\ \varepsilon \end{array}$$

$$\begin{array}{c} \perp, \perp \rightarrow b, b \rightarrow c, c \\ \downarrow \\ \varepsilon, \varepsilon \rightarrow d, d \\ \downarrow \\ \varepsilon, \varepsilon \rightarrow e, e \\ \downarrow \\ \varepsilon, \square \end{array}$$

$$\begin{array}{c} \perp \rightarrow b \rightarrow c \\ \downarrow \\ \varepsilon \rightarrow d \\ \downarrow \\ \varepsilon \rightarrow e \\ \downarrow \\ \varepsilon \end{array}$$

Encoding Example



$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$
 $\perp \quad b$

pop_2

$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε

$\perp, \perp \rightarrow b, b \rightarrow c, c$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow d, d$
 \downarrow
 $\varepsilon, \varepsilon \rightarrow e, e$
 \downarrow
 ε, \square

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε

Encoding Example



TECHNISCHE
UNIVERSITÄT
DARMSTADT

$\perp \quad b \quad c$
 $\perp \quad b \quad d$
 $\perp \quad b \quad d \quad e$
 $\perp \quad b \quad d$

collapse

$\perp \quad b \quad c$

$\perp \rightarrow b \rightarrow c$
 \downarrow
 $\varepsilon \rightarrow d$
 \downarrow
 $\varepsilon \rightarrow e$
 \downarrow
 ε

$\perp, \perp \rightarrow b, b \rightarrow c, c$
 \downarrow
 $\varepsilon, \square \rightarrow d, \square$
 \downarrow
 $\varepsilon, \square \rightarrow e, \square$
 \downarrow
 ε, \square

$\perp \rightarrow b \rightarrow c$

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: Necessarily passed substacks on a run to (q, s)
 2. Identify milestones 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

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

$q_0, \perp \rightarrow$

$q_4, \perp ab$
 $\perp ac$

Milestones

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

$q_0, \perp \rightarrow q_1, \perp a \rightarrow$

$q_4, \perp ab$
 $\perp ac$

Milestones

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

$$q_0, \perp \rightarrow q_1, \perp a \rightarrow q_2, \perp ab \rightarrow \dots$$
$$q_4, \perp ab \rightarrow \perp ac$$

Milestones

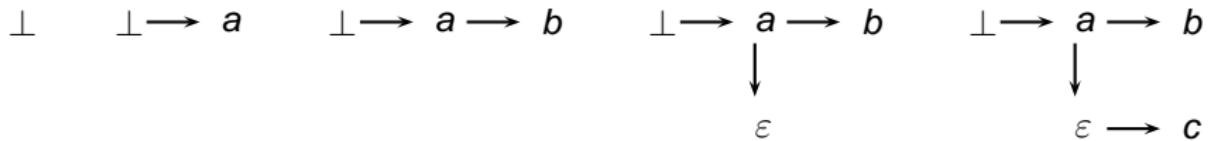
Milestones of s: Necessarily passed substacks on every run to (q, s)

$$q_0, \perp \rightarrow q_1, \perp a \rightarrow q_2, \perp ab \rightarrow q_3, \perp ab \rightarrow q_4, \perp ab \\ \perp a \qquad \qquad \qquad \perp ac$$

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

$$q_0, \perp \rightarrow q_1, \perp a \rightarrow q_2, \perp ab \rightarrow q_3, \perp ab \rightarrow q_4, \perp ab \\ \perp a \qquad \qquad \qquad \perp ac$$

Milestones encoded in the encoding of s :



Example Reachable Configurations



TECHNISCHE
UNIVERSITÄT
DARMSTADT

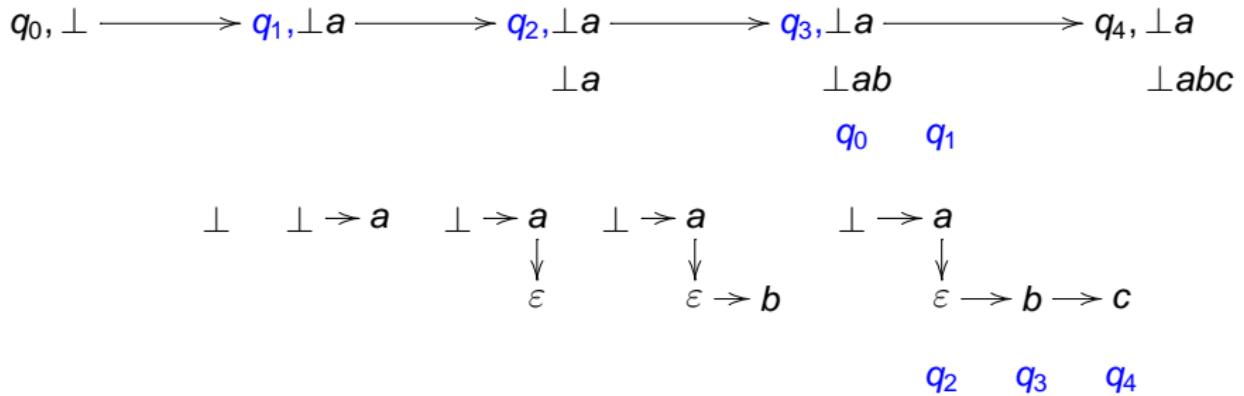
$$\begin{array}{ccccccc} q_0, \perp & \longrightarrow & \perp a & \longrightarrow & \perp a & \longrightarrow & \perp a \\ & & & & \perp a & & \longrightarrow q_4, \perp a \\ & & & & & \perp ab & \\ & & & & & & \perp abc \end{array}$$

$$\begin{array}{ccccccc} \perp & \perp \rightarrow a & \perp \rightarrow a & \perp \rightarrow a & \perp \rightarrow a \\ & & \downarrow \varepsilon & & \downarrow \varepsilon \rightarrow b & & \downarrow \varepsilon \rightarrow b \rightarrow c \end{array}$$

Example Reachable Configurations



TECHNISCHE
UNIVERSITÄT
DARMSTADT



The Loop Problem

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

Definition

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

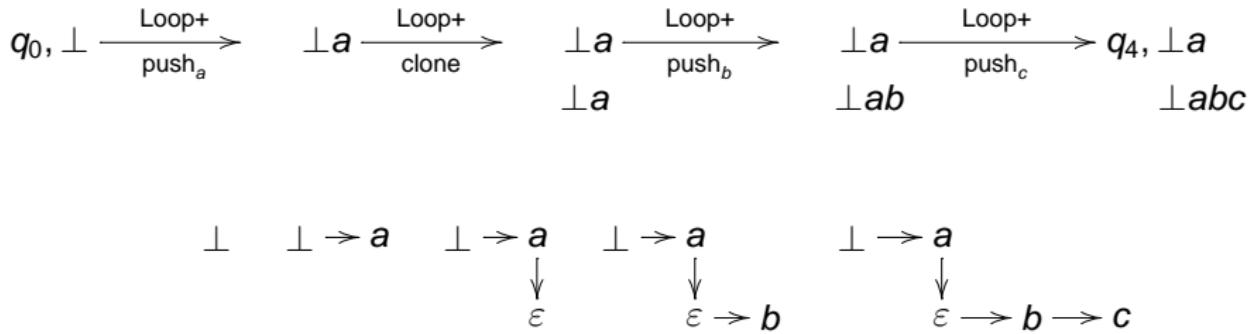
Task for the automaton: Determine $\text{Loops}(s) := \{(q_1, q_2) \text{ s. t. } \exists \text{ loop } q_1, s \text{ to } q_2, s\}$?

Solution:

Lemma (Loop-Lemma)

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

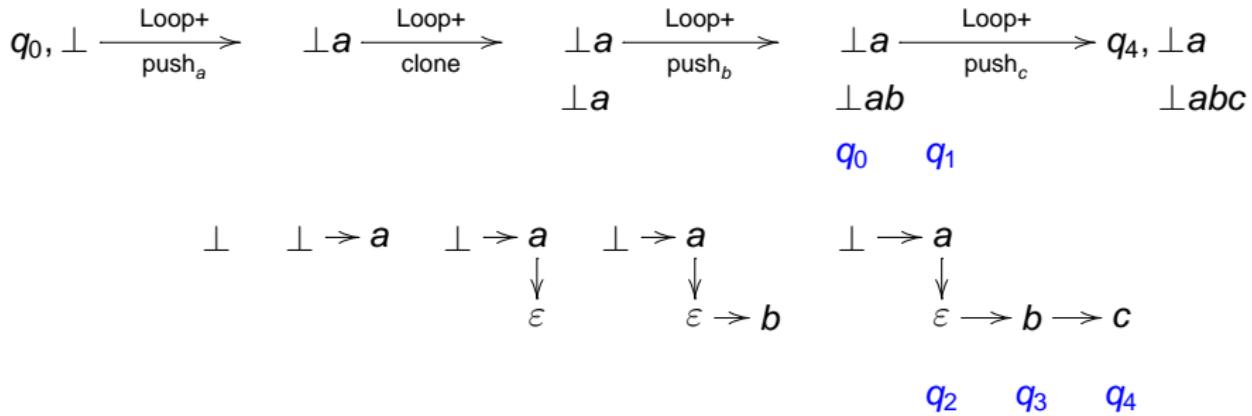
Example Reachable Configurations II



Example Reachable Configurations II

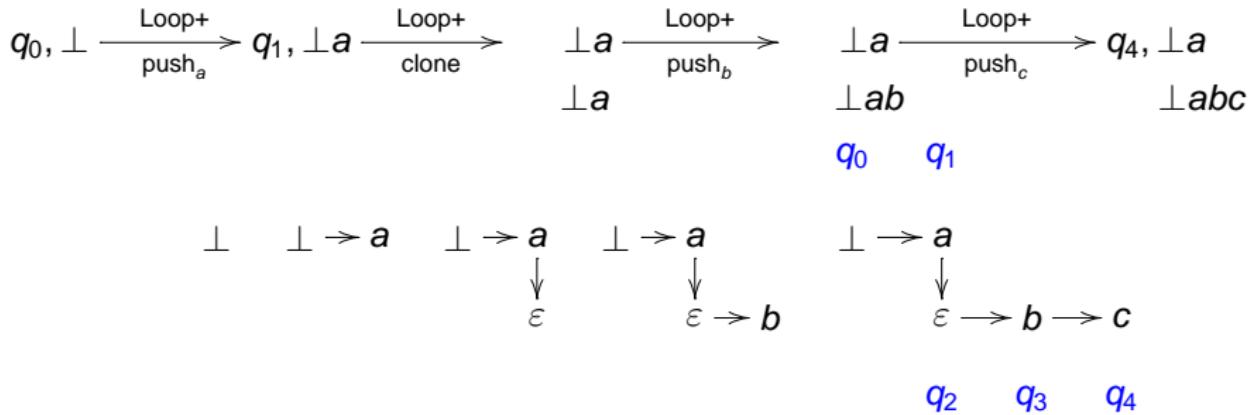


TECHNISCHE
UNIVERSITÄT
DARMSTADT



Labelling valid if

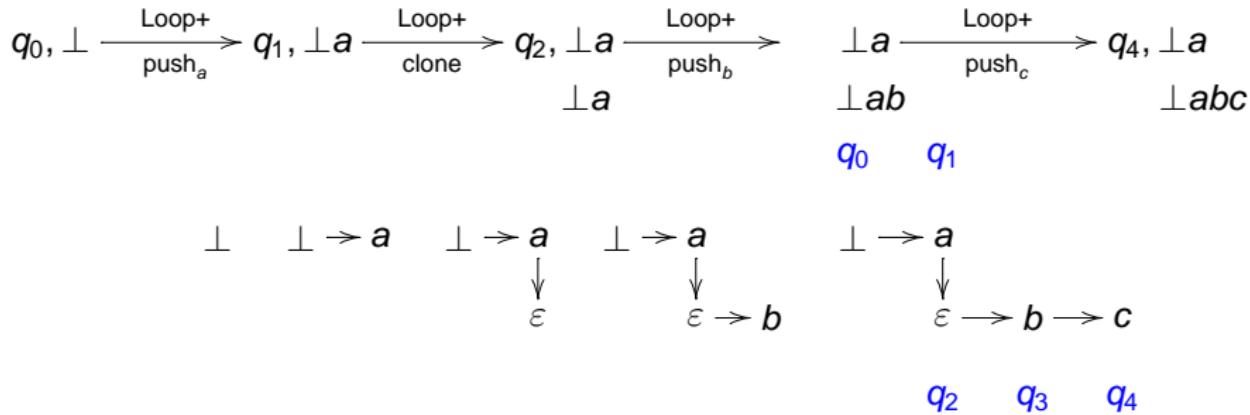
Example Reachable Configurations II



$\exists q'_0 \exists \text{loop}: q_0, \perp \rightarrow q'_0, \perp \text{ and } (q'_0, \perp, q_1, \text{push}_a) \in \Delta$

Labelling valid if

Example Reachable Configurations II

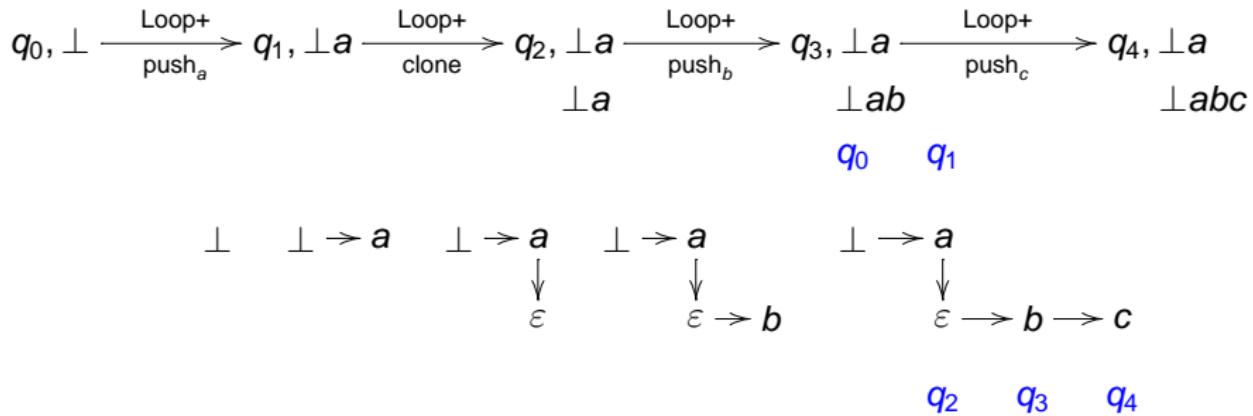


$\exists q'_0 \exists \text{loop}: q_0, \perp \rightarrow q'_0, \perp \text{ and } (q'_0, \perp, q_1, \text{push}_a) \in \Delta$

$\exists q'_1 \exists \text{loop}: q_1, \perp a \rightarrow q'_1, \perp a \text{ and } (q'_1, a, q_2, \text{clone}) \in \Delta$

Labelling valid if

Example Reachable Configurations II



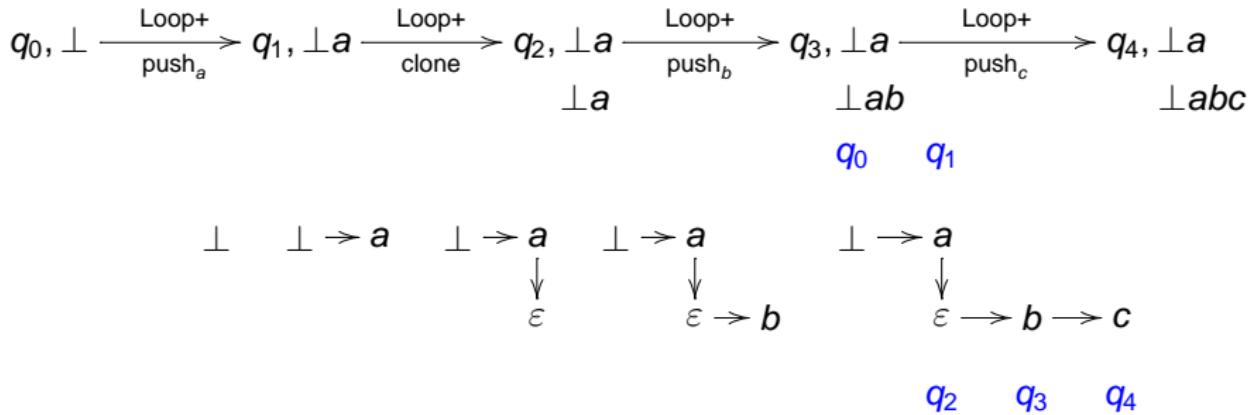
Labelling valid if

$\exists q'_0 \exists \text{loop}: q_0, \perp \rightarrow q'_0, \perp$ and $(q'_0, \perp, q_1, \text{push}_a) \in \Delta$

$\exists q'_1 \exists \text{loop}: q_1, \perp a \rightarrow q'_1, \perp a$ and $(q'_1, a, q_2, \text{clone}) \in \Delta$

$\exists q'_2 \exists \text{loop}: q_2, \perp a \rightarrow q'_2, \perp a$ and $(q'_2, a, q_3, \text{push}_b) \in \Delta$

Example Reachable Configurations II



Labelling valid if

$\exists q'_0 \exists \text{loop}: q_0, \perp \rightarrow q'_0, \perp \text{ and } (q'_0, \perp, q_1, \text{push}_a) \in \Delta$

$\exists q'_1 \exists \text{loop}: q_1, \perp a \rightarrow q'_1, \perp a \text{ and } (q'_1, a, q_2, \text{clone}) \in \Delta$

$\exists q'_2 \exists \text{loop}: q_2, \perp a \rightarrow q'_2, \perp a \text{ and } (q'_2, a, q_3, \text{push}_b) \in \Delta$

$\exists q'_3 \exists \text{loop}: q_3, \perp ab \rightarrow q'_3, \perp ab \text{ and } (q'_3, \perp, q_4, \text{push}_c) \in \Delta$

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)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

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

1. $\text{pop}_1 + \text{Loop of } \text{pop}_1(s) + \text{push}_a$: depends on $\text{Loops}(\text{pop}_1(s))$
2. run reaches s' with topmost word $\text{pop}_1(w) \Rightarrow$ run continues with return:
depends on $\text>Returns(\text{pop}_1(s))$
2. holds because returns and loops only depend on topmost word

Lemma (Return-Lemma)

$\text>Returns(s)$ only depend on $\text>Returns(\text{pop}_1(s))$ and the topmost symbol of s .

Proof.

Same argument as in 2.



Definition

Reach_{xy} 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.



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. □

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?