Regulated Nondeterminism in PDAs: The Non-Regular Case

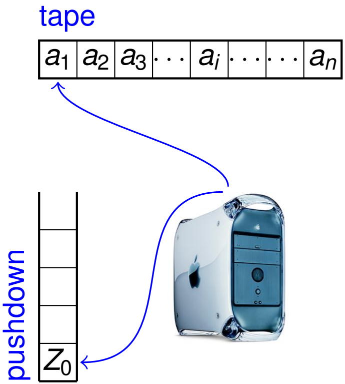
Tomáš Masopust

Faculty of Information Technology
Brno University of Technology
Czech Republic

NCMA 2009

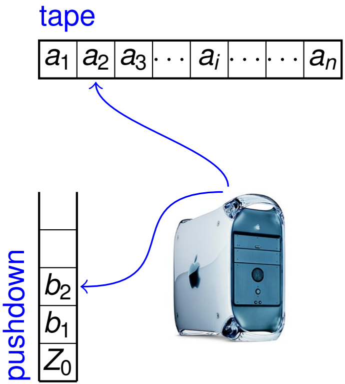
Wroclaw, Poland

Pushdown Automaton



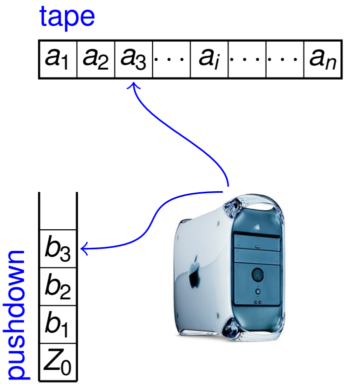
Transitions used:

Pushdown Automaton



Transitions used: r_1

Pushdown Automaton



Transitions used: r_1, r_2

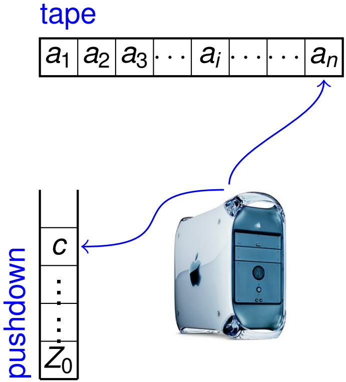
Regulated PDAs (Kolář and Meduna, 2000)

- Motivated by regulations in grammars.
- Given a PDA M and a control language R.

  Transitions used: $r_1, r_2, \ldots, r_k$
- It accepts the input (by a final state) if M accepts the input and $r_1 r_2 \ldots r_k \in R$.

Regulated PDAs (Kolář and Meduna, 2000)

- Motivated by regulations in grammars.
- Given a PDA M and a control language R.

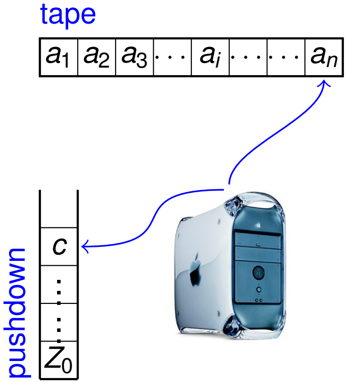


Transitions used: $r_1, r_2, \dots, r_k$

- It accepts the input (by a final state) if M accepts the input and $r_1 r_2 \dots r_k \in R$.

Regulated PDAs (Kolář and Meduna, 2000)

- Motivated by regulations in grammars.
- Given a PDA M and a control language R.



Transitions used: $r_1, r_2, \ldots, r_k$

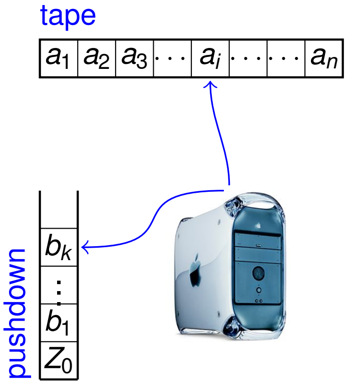
- It accepts the input (by a final state) if M accepts the input and $r_1 r_2 \ldots r_k \in R$.

Regulated PDAs (Kolář and Meduna, 2000)

- Regulated PDAs with regular control languages are ordinary PDAs.
- Regulated PDAs with non-regular (linear) control languages are computationally complete.

Regularly Regulated Pushdowns (Křivka, 2007)

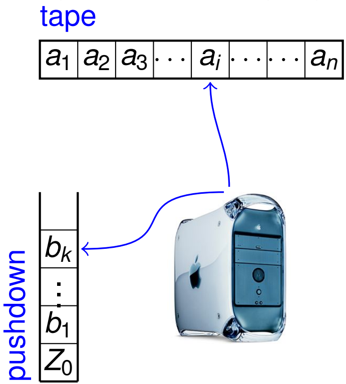
- Given a PDA M and a regular control language R.



- It accepts the input if M accepts and $b_1 \dots b_k \in R$ (in each step).
- Equivalent to ordinary pushdown automata.

Regularly Regulated Pushdowns (Křivka, 2007)

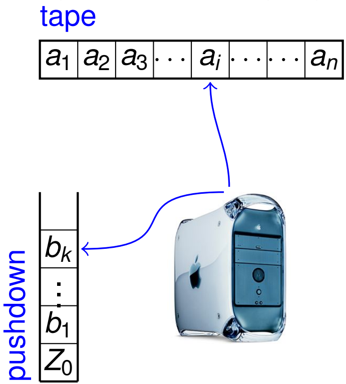
- Given a PDA M and a regular control language R.



- It accepts the input if M accepts and $b_1 \ldots b_k \in R$ (in each step).
- Equivalent to ordinary pushdown automata.

Regularly Regulated Pushdowns (Křivka, 2007)

- Given a PDA M and a regular control language R.



- It accepts the input if M accepts and $b_1 \dots b_k \in R$ (in each step).
- Equivalent to ordinary pushdown automata.

R-PDAs (Kutrib, Malcher, Werlein, 2007)

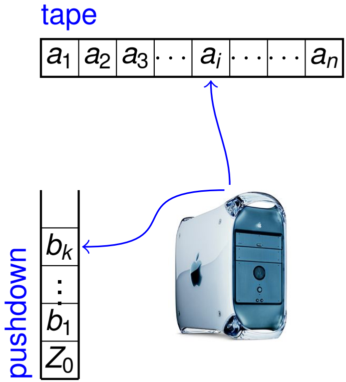
- Generalization: considering nondeterminism.
- Given a PDA M and a control language R

Next step is

$$\begin{cases} b_1 \ldots b_k \in R & \textit{nondeterministic} \\ b_1 \ldots b_k \notin R & \textit{deterministic} \end{cases}$$

R-PDAs (Kutrib, Malcher, Werlein, 2007)

- Generalization: considering nondeterminism.
- Given a PDA M and a control language R



Next step is

$$\begin{cases} b_1 \ldots b_k \in R & \textit{nondeterministic} \\ b_1 \ldots b_k \notin R & \textit{deterministic} \end{cases}$$

Definition

Given a PDA

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

and a control language $R \subseteq (\Gamma \setminus Z_0)^*$. $\mathcal{M}$ is an R-PDA if:

1. for all $q \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $Z \in \Gamma$, δ can be written as

   $$\delta(q, a, Z) = \delta_d(q, a, Z) \cup \delta_{nd}(q, a, Z)\,,$$

   where d = deterministic and nd = nondeterministic, and
2. for all $q, q' \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

   $$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma) \text{ if}$$

   1. either $(q', \gamma') \in \delta_{nd}(q, a, Z)$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \in R$,
   2. or $\delta_d(q, a, Z) = (q', \gamma')$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \notin R$.

Definition

Given a PDA

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

and a control language $R \subseteq (\Gamma \setminus Z_0)^*$. $\mathcal{M}$ is an R-PDA if:

1. for all $q \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $Z \in \Gamma$, δ can be written as

$$\delta(q, a, Z) = \delta_d(q, a, Z) \cup \delta_{nd}(q, a, Z)\,,$$

where $d =$ deterministic and $nd =$ nondeterministic, and

2. for all $q, q' \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

$$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma) \text{ if}$$

   1. either $(q', \gamma') \in \delta_{nd}(q, a, Z)$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \in R$,
   2. or $\delta_d(q, a, Z) = (q', \gamma')$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \notin R$.

Definition

Given a PDA

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

and a control language $R \subseteq (\Gamma \setminus Z_0)^*$. $\mathcal{M}$ is an R-PDA if:

1. for all $q \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $Z \in \Gamma$, δ can be written as

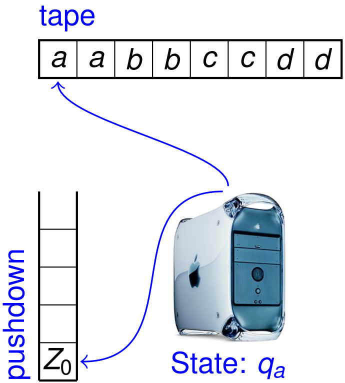
   $$\delta(q, a, Z) = \delta_d(q, a, Z) \cup \delta_{nd}(q, a, Z)\,,$$

   where $d =$ deterministic and $nd =$ nondeterministic, and
2. for all $q, q' \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

   $$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma) \text{ if}$$

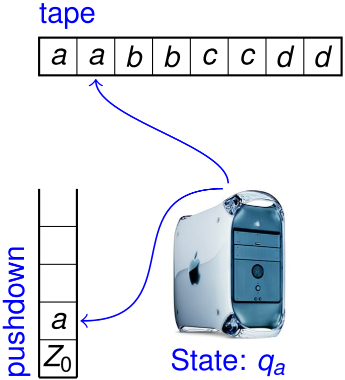
   1. either $(q', \gamma') \in \delta_{nd}(q, a, Z)$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \in R$,
   2. or $\delta_d(q, a, Z) = (q', \gamma')$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \notin R$.

Example – $R = \{a^n b^n : n \geq 1\}$



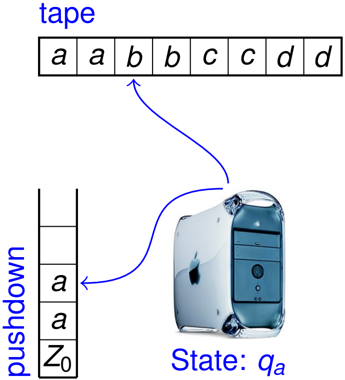
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



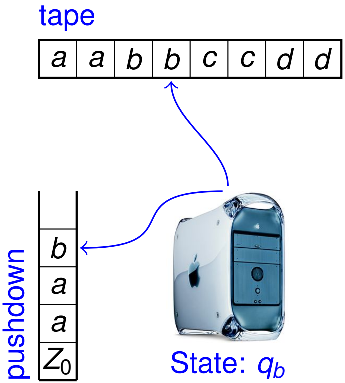
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



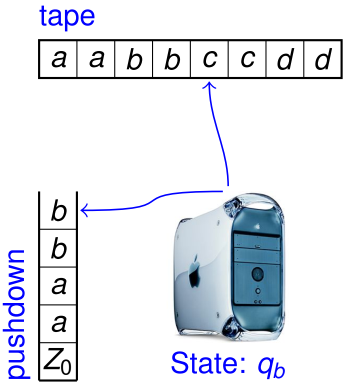
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



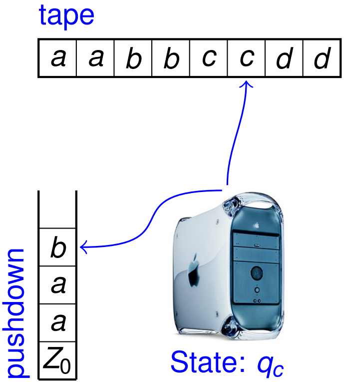
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



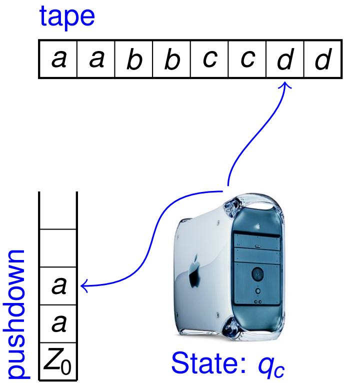
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



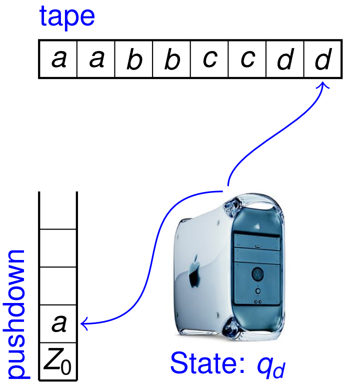
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



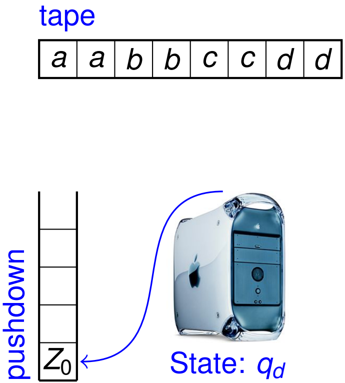
- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$



- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$

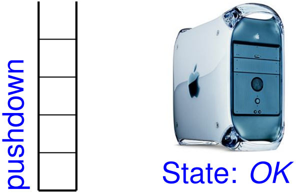


- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

Example – $R = \{a^n b^n : n \geq 1\}$

tape

a	a	b	b	c	c	d	d



- $\mathcal{M}$ nondeterministically checks that $Z_0 a^m b^n \in R$, i.e., $m = n$.

$$T(\mathcal{M}) = \{a^n b^n c^n d^n : n \geq 1\}.$$

Properties

- R-PDAs behave nondeterministically iff their pushdown content forms a string belonging to R.
- R is regular, then the power of PDAs.

Theorem

Let R be a regular language and $\mathcal{M}$ be an R-PDA. Then, an equivalent PDA $\mathcal{M}'$ can effectively be constructed.

- R is linear, then the power increases.
- What is the power of R-PDAs with non-regular control languages?

Properties

- R-PDAs behave nondeterministically iff their pushdown content forms a string belonging to R.
- R is regular, then the power of PDAs.

Theorem

Let R be a regular language and $\mathcal{M}$ be an R-PDA. Then, an equivalent PDA $\mathcal{M}'$ can effectively be constructed.

- R is linear, then the power increases.
- What is the power of R-PDAs with non-regular control languages?

Properties

- R-PDAs behave nondeterministically iff their pushdown content forms a string belonging to R.
- R is regular, then the power of PDAs.

Theorem

Let R be a regular language and $\mathcal{M}$ be an R-PDA. Then, an equivalent PDA $\mathcal{M}'$ can effectively be constructed.

- R is linear, then the power increases.
- What is the power of R-PDAs with non-regular control languages?

Properties

- R-PDAs behave nondeterministically iff their pushdown content forms a string belonging to R.
- R is regular, then the power of PDAs.

Theorem

Let R be a regular language and $\mathcal{M}$ be an R-PDA. Then, an equivalent PDA $\mathcal{M}'$ can effectively be constructed.

- R is linear, then the power increases.
- What is the power of R-PDAs with non-regular control languages?

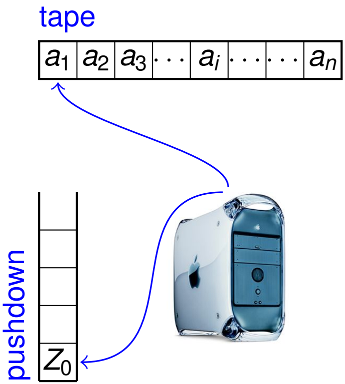
R-PDAs: The Non-Regular Case

Theorem

Let $L \in RE$. Then, there is a linear language R and an R-PDA $\mathcal{M}$ such that

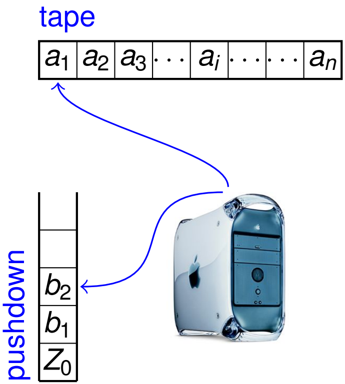
$$L = T(\mathcal{M}) .$$

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



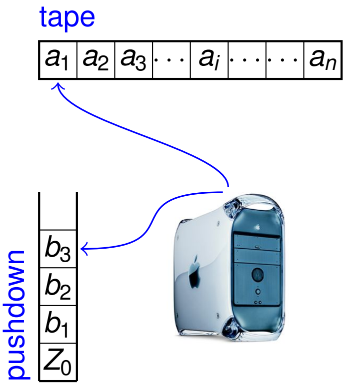
- $\mathcal{M}$ nondeterministically pushes symbols onto its pushdown.

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



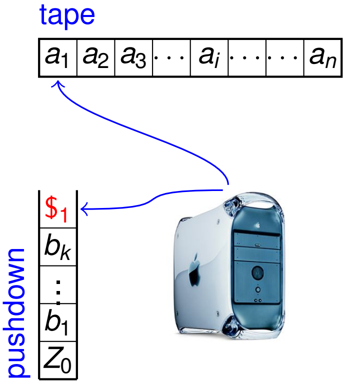
- $\mathcal{M}$ nondeterministically pushes symbols onto its pushdown.

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



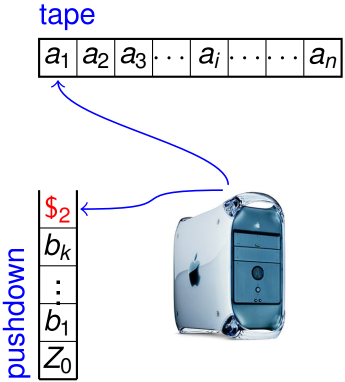
- $\mathcal{M}$ nondeterministically pushes symbols onto its pushdown.

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



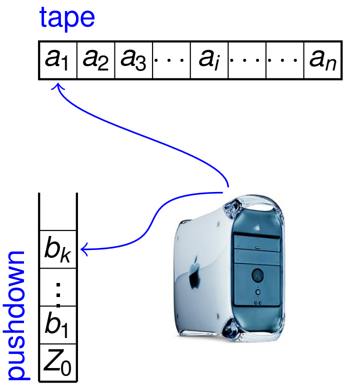
- $\gamma\$_1 \in L_1\$_1$?

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



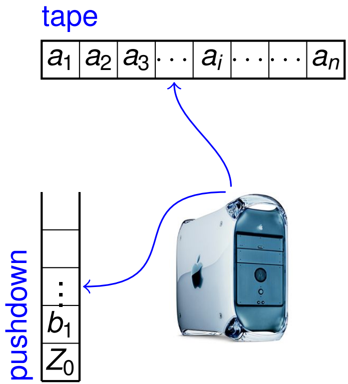
- $\gamma\$_1 \in L_1\$_1$ – YES; $\gamma\$_2 \in L_2\$_2$?

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



- $\gamma\$_1 \in L_1\$_1$ – YES; $\gamma\$_2 \in L_2\$_2$ – YES; we have $\gamma \in L_1 \cap L_2$.

Proof (sketch) – $L^R = h(L_1 \cap L_2)$, L_1, L_2 linear



- Remove b_k, read $h(b_k)^R$ from the input.

Corollary

Let $L \in RE$. Then, there is a linear (deterministic) context-free language R and an R-PDA

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

such that $L = T(\mathcal{M})$,

- $|Q| \leq 3$,
- $|\Gamma| \leq |\Sigma| + 7$.

State-Controlled R-PDAs (PDAs with an oracle)

Let

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_c, Z_0, F)$$

be a PDA, where $Q_c \subseteq Q$ is a set of checking states. $R \subseteq (\Gamma \setminus Z_0)^*$.

$\mathcal{M}$ is called a state-controlled R-PDA (R-sPDA) if for all $q, q' \in Q$, $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $Z \in \Gamma$, and $\gamma \in \Gamma^*$,

$$(q, aw, Z\gamma) \vdash_{\mathcal{M}} (q', w, \gamma'\gamma)$$

if $(q', \gamma') \in \delta(q, a, Z)$ and

1. either $q \in Q \setminus Q_c$,
2. or $q \in Q_c$, $Z\gamma = \gamma'' Z_0$, and $(\gamma'')^R \in R$.

Power

Theorem

Let R be a regular language and $\mathcal{M}$ be an R-sPDA. Then, an equivalent PDA $\mathcal{M}'$ can effectively be constructed.

Power

Theorem

Let R be a regular language and $\mathcal{M}$ be an R-sPDA. Then, an equivalent PDA $\mathcal{M}'$ can effectively be constructed.

Theorem

Let $L \in RE$. Then, there is a linear language R and an R-sPDA $\mathcal{M}$ such that

$$L = T(\mathcal{M}) .$$

In addition, $\mathcal{M}$ checks the pushdown content no more than twice during any computation.

Corollary

Let $L \in RE$. Then, there is a linear language R and an R-sPDA

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_c, Z_0, F)$$

which checks the pushdown content no more than twice during any computation, such that

- $|Q| \leq 4$,
- $|Q_c| = 1$,
- $|\Gamma| \leq |\Sigma| + 6$,

and $L = T(\mathcal{M})$.

Open Problems

By the example, there is a (deterministic) R-sPDA $\mathcal{M}$, where

- $R = \{a^n b^n : n \geq 1\}$ is linear, deterministic context-free,
- $T(\mathcal{M}) = \{a^n b^n c^n d^n : n \geq 1\}$,
- only one check of the pushdown content.

Open Problems

By the example, there is a (deterministic) R-sPDA $\mathcal{M}$, where

- $R = \{a^n b^n : n \geq 1\}$ is linear, deterministic context-free,
- $T(\mathcal{M}) = \{a^n b^n c^n d^n : n \geq 1\}$,
- only one check of the pushdown content.

Open Problem

What is the power of R-sPDAs with one check of the pushdown content and R linear?

Open Problems

By the example, there is a (deterministic) R-sPDA $\mathcal{M}$, where

- $R = \{a^n b^n : n \geq 1\}$ is linear, deterministic context-free,
- $T(\mathcal{M}) = \{a^n b^n c^n d^n : n \geq 1\}$,
- only one check of the pushdown content.

Open Problem

What is the power of R-sPDAs with one check of the pushdown content and R linear?

Open Problem

What is the power of deterministic R-sPDAs with R linear?

Open Problems (Deterministic R-sPDAs)

- Deterministic R-sPDAs (R-sDPDAs), R linear.
- $DCF \subset R\text{-sDPDA} \subseteq REC \quad (CS, detCS)$.
- Is $CF \subseteq R\text{-sDPDA}$?

Open Problems (Deterministic R-sPDAs)

- Deterministic R-sPDAs (R-sDPDAs), R linear.
- $DCF \subset R\text{-sDPDA} \subseteq REC \quad (CS, detCS)$.
- Is $CF \subseteq R\text{-sDPDA}$?

Open Problems (Deterministic R-sPDAs)

- Deterministic R-sPDAs (R-sDPDAs), R linear.
- $DCF \subset R\text{-sDPDA} \subseteq REC \quad (CS, detCS)$.
- Is $CF \subseteq R\text{-sDPDA}$?

Open Problems (R-PDAs, R-sPDAs, etc.)

- Closure properties: fix R and two R-PDAs (R-sPDAs) $\mathcal{M}_1$ and $\mathcal{M}_2$. Are the languages
  - $T(\mathcal{M}_1) \cup T(\mathcal{M}_2)$
  - $T(\mathcal{M}_1) \cap T(\mathcal{M}_2)$
  - $T(\mathcal{M}_1) \cdot T(\mathcal{M}_2)$
  - $T(\mathcal{M}_1)^*$ etc.

  accepted by an R-PDA (R-sPDA)?
- Decidable/undecidable problems.

Open Problems (R-PDAs, R-sPDAs, etc.)

- Closure properties: fix R and two R-PDAs (R-sPDAs) $\mathcal{M}_1$ and $\mathcal{M}_2$. Are the languages
  - $T(\mathcal{M}_1) \cup T(\mathcal{M}_2)$
  - $T(\mathcal{M}_1) \cap T(\mathcal{M}_2)$
  - $T(\mathcal{M}_1) \cdot T(\mathcal{M}_2)$
  - $T(\mathcal{M}_1)^*$ etc.

  accepted by an R-PDA (R-sPDA)?
- Decidable/undecidable problems.

Thank You