Input-Driven Stack Automata

Suna Bensch Markus Holzer Martin Kutrib
Andreas Malcher

Department of Computing Science, Umeå University, Sweden
Institut für Informatik, Universität Gießen, Germany

IFIP TCS 2012, Amsterdam, Holland

Input-driven automata

- The input alphabet is divided into several classes.

Input-driven automata

- The input alphabet is divided into several classes.
- Each class induces a specific behavior of the automaton.

Input-driven automata

- The input alphabet is divided into several classes.
- Each class induces a specific behavior of the automaton.

Example: Input-driven pushdown automata

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$$

Input-driven automata

- The input alphabet is divided into several classes.
- Each class induces a specific behavior of the automaton.

Example: Input-driven pushdown automata

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store

Input-driven automata

- The input alphabet is divided into several classes.
- Each class induces a specific behavior of the automaton.

Example: Input-driven pushdown automata

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store
- $a \in \Sigma_r$: a symbol is popped from the pushdown store

Input-driven automata

- The input alphabet is divided into several classes.
- Each class induces a specific behavior of the automaton.

Example: Input-driven pushdown automata

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store
- $a \in \Sigma_r$: a symbol is popped from the pushdown store
- $a \in \Sigma_i$: internal change of states, no action on the pushdown store

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)
  - nested word automata

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)
  - nested word automata
  - $2^{\Omega(n^2)}$ bounds for determinization

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)
  - nested word automata
  - $2^{\Omega(n^2)}$ bounds for determinization
  - closure properties, decidability questions

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)
  - nested word automata
  - $2^{\Omega(n^2)}$ bounds for determinization
  - closure properties, decidability questions
- Pushdown forest automata (Neumann, Seidl 1998; Gauwin, Niehren, Roos 2008)

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)
  - nested word automata
  - $2^{\Omega(n^2)}$ bounds for determinization
  - closure properties, decidability questions
- Pushdown forest automata (Neumann, Seidl 1998; Gauwin, Niehren, Roos 2008)
- Descriptional complexity aspects (Han, Salomaa 2009, Piao, Salomaa 2009, Okhotin, Salomaa 2011)

Input-driven pushdown automata

- Input-driven PDA accept context-free languages in $O(\log n)$ space. (Mehlhorn 1980, von Braunmühl, Verbeek 1983)
- Nondeterministic and deterministic versions are equivalent. (von Braunmühl, Verbeek 1983)
- Input-driven languages are in NC^1. (Dymond 1988)
- Visibly pushdown automata (Alur, Madhusudan 2004)
  - nested word automata
  - $2^{\Omega(n^2)}$ bounds for determinization
  - closure properties, decidability questions
- Pushdown forest automata (Neumann, Seidl 1998; Gauwin, Niehren, Roos 2008)
- Descriptional complexity aspects (Han, Salomaa 2009, Piao, Salomaa 2009, Okhotin, Salomaa 2011)
- Extensions/generalizations: multiple stacks, graph automata, height-deterministic PDA, ...

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines
- Idea: Allow up- and down-moves inside the pushdown store.

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines
- Idea: Allow up- and down-moves inside the pushdown store.
- Modifications are only allowed on top of the pushdown store.

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines
- Idea: Allow up- and down-moves inside the pushdown store.
- Modifications are only allowed on top of the pushdown store.
- Non-erasing variants: no pop-moves are allowed.

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines
- Idea: Allow up- and down-moves inside the pushdown store.
- Modifications are only allowed on top of the pushdown store.
- Non-erasing variants: no pop-moves are allowed.
- Deterministic and nondeterministic variants.

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines
- Idea: Allow up- and down-moves inside the pushdown store.
- Modifications are only allowed on top of the pushdown store.
- Non-erasing variants: no pop-moves are allowed.
- Deterministic and nondeterministic variants.
- One-way and two-way variants.

Stack automata

- Introduced by Ginsburg, Greibach, and Harrison in 1967.
- Motivation: compiling issues, intermediate model between PDA and Turing machines
- Idea: Allow up- and down-moves inside the pushdown store.
- Modifications are only allowed on top of the pushdown store.
- Non-erasing variants: no pop-moves are allowed.
- Deterministic and nondeterministic variants.
- One-way and two-way variants.
- Decidability for one-way stack automata: emptiness is decidable for nondeterministic machines, equivalence with regular sets is decidable for deterministic machines.

Input-driven stack automata

$$M = \langle Q, \Sigma, \Gamma, \bot, q_0, F, \delta_c, \delta_r, \delta_i, \delta_d, \delta_u \rangle,$$

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i \cup \Sigma_d \cup \Sigma_u$$

Input-driven stack automata

$M = \langle Q, \Sigma, \Gamma, \bot, q_0, F, \delta_c, \delta_r, \delta_i, \delta_d, \delta_u \rangle,$

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i \cup \Sigma_d \cup \Sigma_u$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store

Input-driven stack automata

$$M = \langle Q, \Sigma, \Gamma, \bot, q_0, F, \delta_c, \delta_r, \delta_i, \delta_d, \delta_u \rangle,$$

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i \cup \Sigma_d \cup \Sigma_u$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store
- $a \in \Sigma_r$: a symbol is popped from the pushdown store

Input-driven stack automata

$$M = \langle Q, \Sigma, \Gamma, \bot, q_0, F, \delta_c, \delta_r, \delta_i, \delta_d, \delta_u \rangle,$$

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i \cup \Sigma_d \cup \Sigma_u$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store
- $a \in \Sigma_r$: a symbol is popped from the pushdown store
- $a \in \Sigma_i$: internal change of states, no action on the pushdown store

Input-driven stack automata

$$M = \langle Q, \Sigma, \Gamma, \bot, q_0, F, \delta_c, \delta_r, \delta_i, \delta_d, \delta_u \rangle,$$

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i \cup \Sigma_d \cup \Sigma_u$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store
- $a \in \Sigma_r$: a symbol is popped from the pushdown store
- $a \in \Sigma_i$: internal change of states, no action on the pushdown store
- $a \in \Sigma_d$: down-move of the pushdown store pointer

Input-driven stack automata

$$M = \langle Q, \Sigma, \Gamma, \bot, q_0, F, \delta_c, \delta_r, \delta_i, \delta_d, \delta_u \rangle,$$

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i \cup \Sigma_d \cup \Sigma_u$$

- $a \in \Sigma_c$: a symbol is pushed onto the pushdown store
- $a \in \Sigma_r$: a symbol is popped from the pushdown store
- $a \in \Sigma_i$: internal change of states, no action on the pushdown store
- $a \in \Sigma_d$: down-move of the pushdown store pointer
- $a \in \Sigma_u$: up-move of the pushdown store pointer

Example

$\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\text{1sNEDVSA})$.

Example

$\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\text{1sNEDVSA})$.

- $\Sigma_c = \{a\}$, $\Sigma_r = \{\}$, $\Sigma_i = \{\}$, $\Sigma_u = \{c\}$, $\Sigma_d = \{b\}$

Example

$\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\text{1sNEDVSA})$.

- $\Sigma_c = \{a\}$, $\Sigma_r = \{\}$, $\Sigma_i = \{\}$, $\Sigma_u = \{c\}$, $\Sigma_d = \{b\}$
- While reading a's, push some symbol A on the pushdown store.

Example

$\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{1sNEDVSA})$.

- $\Sigma_c = \{a\}$, $\Sigma_r = \{\}$, $\Sigma_i = \{\}$, $\Sigma_u = \{c\}$, $\Sigma_d = \{b\}$
- While reading a's, push some symbol A on the pushdown store.
- While reading b's, move downwards up to $\perp$.

Example

$\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{1sNEDVSA})$.

- $\Sigma_c = \{a\}$, $\Sigma_r = \{\}$, $\Sigma_i = \{\}$, $\Sigma_u = \{c\}$, $\Sigma_d = \{b\}$
- While reading a's, push some symbol A on the pushdown store.
- While reading b's, move downwards up to $\perp$.
- While reading c's, move upwards up to the top and accept in an additional time step.

Example

$\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{1sNEDVSA})$.

- $\Sigma_c = \{a\}$, $\Sigma_r = \{\}$, $\Sigma_i = \{\}$, $\Sigma_u = \{c\}$, $\Sigma_d = \{b\}$
- While reading a's, push some symbol A on the pushdown store.
- While reading b's, move downwards up to $\bot$.
- While reading c's, move upwards up to the top and accept in an additional time step.

We consider also weak variants of input-driven stack automata. Here, the moves inside the pushdown store don't have to be input-driven.

Comparing weak and strong variants

Lemma

$L_1 = \{ a^n b a^{n-1} c \mid n \geq 1 \}$ belongs to $\mathscr{L}(1\text{wNEDVSA})$, but is not accepted by any 1sDVSA.

Comparing weak and strong variants

Lemma

$L_1 = \{\, a^n b a^{n-1} c \mid n \geq 1 \,\}$ belongs to $\mathscr{L}(1\mathrm{wNEDVSA})$, but is not accepted by any $1\mathrm{sDVSA}$.

- While reading a's, push some symbol A on the pushdown store.

Comparing weak and strong variants

Lemma

$L_1 = \{\, a^n b a^{n-1} c \mid n \geq 1 \,\}$ belongs to $\mathscr{L}(1\mathrm{wNEDVSA})$, but is not accepted by any $1\mathrm{sDVSA}$.

- While reading a's, push some symbol A on the pushdown store.
- When reading b, move downwards and enter some other state.

Comparing weak and strong variants

Lemma

$L_1 = \{\, a^n b a^{n-1} c \mid n \geq 1 \,\}$ belongs to $\mathscr{L}(1\mathrm{wNEDVSA})$, but is not accepted by any $1\mathrm{sDVSA}$.

- While reading a's, push some symbol A on the pushdown store.
- When reading b, move downwards and enter some other state.
- While reading a's, move downwards.

Comparing weak and strong variants

Lemma

$L_1 = \{\, a^n b a^{n-1} c \mid n \geq 1 \,\}$ belongs to $\mathscr{L}(1\mathrm{wNEDVSA})$, but is not accepted by any $1\mathrm{sDVSA}$.

- While reading a's, push some symbol A on the pushdown store.
- When reading b, move downwards and enter some other state.
- While reading a's, move downwards.
- When reading c and seeing $\perp$, move upwards and accept.

Comparing weak and strong variants

Lemma

$L_1 = \{ a^n b a^{n-1} c \mid n \geq 1 \}$ belongs to $\mathscr{L}(1\text{wNEDVSA})$, but is not accepted by any 1sDVSA.

- ➜ While reading a's, push some symbol A on the pushdown store.
- ➜ When reading b, move downwards and enter some other state.
- ➜ While reading a's, move downwards.
- ➜ When reading c and seeing $\perp$, move upwards and accept.

It can be shown by pumping arguments that L_1 is not accepted by any 1sDVSA.

Comparison with context-free languages

Lemma

- $\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{1sNEDVSA})$.

Comparison with context-free languages

Lemma

- $\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{1sNEDVSA})$.
- $L_2 = \{ a^n b^m a b^m a^n \mid n, m \geq 1 \} \cup \{ b^n a^n \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{DCFL})$, but is not accepted by any $\mathrm{1wDVSA}$.

Comparison with context-free languages

Lemma

- $\{ a^n b^n c^{n+1} \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{1sNEDVSA})$.
- $L_2 = \{ a^n b^m a b^m a^n \mid n, m \geq 1 \} \cup \{ b^n a^n \mid n \geq 1 \}$ belongs to $\mathscr{L}(\mathrm{DCFL})$, but is not accepted by any $\mathrm{1wDVSA}$.
- Every VPDA can be simulated by some equivalent $\mathrm{1sDVSA}$.

Comparing erasing and non-erasing variants

Lemma

$L_3 = \{ a^{n+m} b^m w \hat{w}^R b^n \mid m, n \geq 1, w \in \{0,1\}^* \}$ belongs to $\mathscr{L}(\mathrm{1sDVSA})$, but is not accepted by any $\mathrm{1wNEDVSA}$.

Comparing erasing and non-erasing variants

Lemma

$L_3 = \{ a^{n+m} b^m w \hat{w}^R b^n \mid m, n \geq 1, w \in \{0,1\}^* \}$ belongs to $\mathscr{L}(\mathrm{1sDVSA})$, but is not accepted by any $\mathrm{1wNEDVSA}$.

- L_3 can be accepted by some VPDA and thus by some $\mathrm{1sDVSA}$.

Comparing erasing and non-erasing variants

Lemma

$L_3 = \{\, a^{n+m} b^m w \hat{w}^R b^n \mid m, n \geq 1, w \in \{0,1\}^* \,\}$ belongs to $\mathscr{L}(\mathrm{1sDVSA})$, but is not accepted by any $\mathrm{1wNEDVSA}$.

- L_3 can be accepted by some VPDA and thus by some $\mathrm{1sDVSA}$.
- With incompressibility arguments from the field of Kolmogorov complexity, it can be shown that $L_3 \notin \mathscr{L}(\mathrm{1wNEDVSA})$.

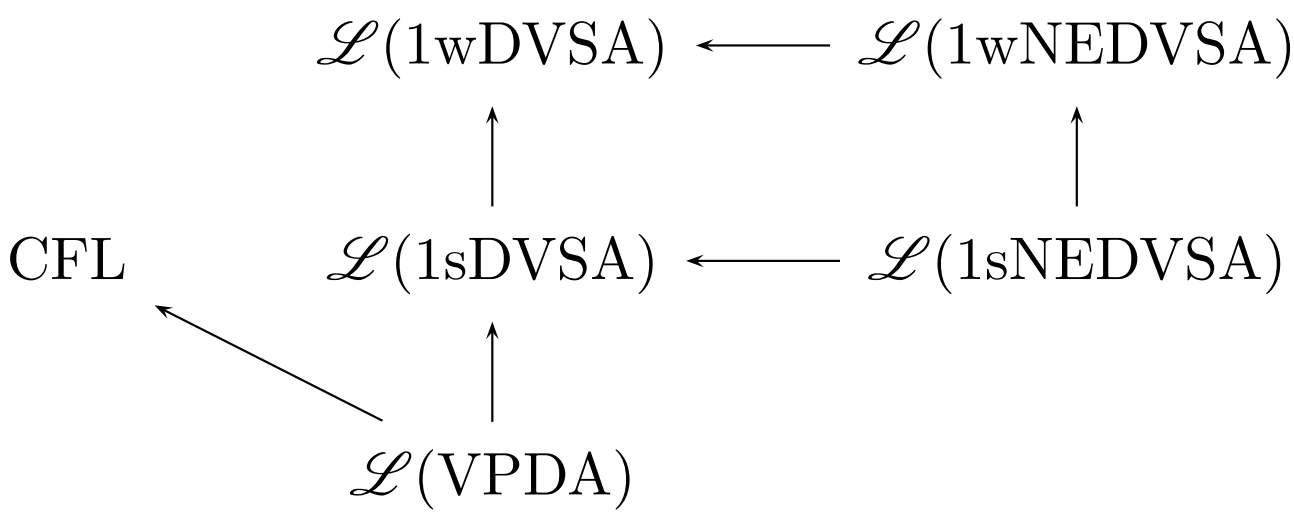
Comparing erasing and non-erasing variants

Lemma

$L_3 = \{\, a^{n+m} b^m w \hat{w}^R b^n \mid m, n \geq 1, w \in \{0,1\}^* \,\}$ belongs to $\mathscr{L}(\mathrm{1sDVSA})$, but is not accepted by any $\mathrm{1wNEDVSA}$.

- L_3 can be accepted by some VPDA and thus by some $\mathrm{1sDVSA}$.
- With incompressibility arguments from the field of Kolmogorov complexity, it can be shown that $L_3 \notin \mathscr{L}(\mathrm{1wNEDVSA})$.
- L_3 is a context-free language where erasing is essential.

Summary for deterministic machines



Comparing deterministic and nondeterministic variants

$$L_4 = T_1 \cup T_2 \cup T_1' \cup T_2' \cup T_3 \cup T_4$$

Comparing deterministic and nondeterministic variants

$$L_4 = T_1 \cup T_2 \cup T_1' \cup T_2' \cup T_3 \cup T_4$$

$$T_1 = \{\, a^n d_1^n u_1^{n+1} \mid n \geq 1 \,\}$$
$$T_2 = \{\, a^n d_2^n u_2^{n+1} \mid n \geq 1 \,\}$$
$$T_1' = \{\, a^n u_1^m d_1^n u_1^{n+1} \mid m, n \geq 1 \,\}$$
$$T_2' = \{\, a^n u_2^m d_2^n u_2^{n+1} \mid m, n \geq 1 \,\}$$
$$T_3 = \{\, a^n d_1^{n+1} \mid n \geq 1 \,\}$$
$$T_4 = \{\, a^{n+m} d_1^m d_2^n u_2^{n+1} u_1^m w \hat{w}^R d_1^m d_2^{n+1} \mid m, n \geq 1, w \in \{0,1\}^+ \,\}$$

Comparing deterministic and nondeterministic variants

$$L_4 = T_1 \cup T_2 \cup T_1' \cup T_2' \cup T_3 \cup T_4$$

$$T_1 = \{\, a^n d_1^m u_1^{n+1} \mid n \geq 1 \,\}$$
$$T_2 = \{\, a^n d_2^m u_2^{n+1} \mid n \geq 1 \,\}$$
$$T_1' = \{\, a^n u_1^m d_1^n u_1^{n+1} \mid m, n \geq 1 \,\}$$
$$T_2' = \{\, a^n u_2^m d_2^n u_2^{n+1} \mid m, n \geq 1 \,\}$$
$$T_3 = \{\, a^n d_1^{n+1} \mid n \geq 1 \,\}$$
$$T_4 = \{\, a^{n+m} d_1^m d_2^n u_2^{n+1} u_1^m w \hat{w}^R d_1^m d_2^{n+1} \mid m, n \geq 1, w \in \{0,1\}^+ \,\}$$

Lemma

L_4 belongs to $\mathscr{L}(\text{1sNENVSA})$, but is not accepted by any 1wDVSA.

Comparing deterministic and nondeterministic variants

It is possible to separate all considered deterministic classes from there nondeterministic counterparts.

Corollary

- $\mathscr{L}(\mathrm{1sNEDVSA}) \subset \mathscr{L}(\mathrm{1sNENVSA})$.
- $\mathscr{L}(\mathrm{1sDVSA}) \subset \mathscr{L}(\mathrm{1sNVSA})$.
- $\mathscr{L}(\mathrm{1wNEDVSA}) \subset \mathscr{L}(\mathrm{1wNENVSA})$.
- $\mathscr{L}(\mathrm{1wDVSA}) \subset \mathscr{L}(\mathrm{1wNVSA})$.

Comparing deterministic and nondeterministic variants

It is possible to separate all considered deterministic classes from there nondeterministic counterparts.

Corollary

- $\mathscr{L}(\mathrm{1sNEDVSA}) \subset \mathscr{L}(\mathrm{1sNENVSA})$.
- $\mathscr{L}(\mathrm{1sDVSA}) \subset \mathscr{L}(\mathrm{1sNVSA})$.
- $\mathscr{L}(\mathrm{1wNEDVSA}) \subset \mathscr{L}(\mathrm{1wNENVSA})$.
- $\mathscr{L}(\mathrm{1wDVSA}) \subset \mathscr{L}(\mathrm{1wNVSA})$.

This is a large difference to the situation for VPDA.

Consequences for the computational complexity

Fixed membership problem: given a fixed automaton M and some input w. Decide whether or not w is accepted by M.

Consequences for the computational complexity

Fixed membership problem: given a fixed automaton M and some input w. Decide whether or not w is accepted by M.

The fixed membership problem is

- ➜ LOGCFL-complete for PDA,

Consequences for the computational complexity

Fixed membership problem: given a fixed automaton M and some input w. Decide whether or not w is accepted by M.

The fixed membership problem is

- ➜ LOGCFL-complete for PDA,
- ➜ NC^1-complete for VPDA,

Consequences for the computational complexity

Fixed membership problem: given a fixed automaton M and some input w. Decide whether or not w is accepted by M.

The fixed membership problem is

- ➜ LOGCFL-complete for PDA,
- ➜ NC^1-complete for VPDA,
- ➜ P-complete for one-way deterministic stack automata,

Consequences for the computational complexity

Fixed membership problem: given a fixed automaton M and some input w. Decide whether or not w is accepted by M.

The fixed membership problem is

- ➜ LOGCFL-complete for PDA,
- ➜ NC^1-complete for VPDA,
- ➜ P-complete for one-way deterministic stack automata,
- ➜ NP-complete for one-way nondeterministic stack automata.

Consequences for the computational complexity

Fixed membership problem: given a fixed automaton M and some input w. Decide whether or not w is accepted by M.

The fixed membership problem is

- ➜ LOGCFL-complete for PDA,
- ➜ NC^1-complete for VPDA,
- ➜ P-complete for one-way deterministic stack automata,
- ➜ NP-complete for one-way nondeterministic stack automata.

Theorem

Each of the language families $\mathscr{L}(\mathrm{1sNENVSA})$, $\mathscr{L}(\mathrm{1sNVSA})$, $\mathscr{L}(\mathrm{1wNENVSA})$, and $\mathscr{L}(\mathrm{1wNVSA})$ has an NP-complete fixed membership problem.

Summary

Summary

- We have introduced input-driven stack automata.

Summary

- We have introduced input-driven stack automata.
- Nondeterministic variants are more powerful than deterministic variants.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.
- ➜ Fixed membership problem remains NP-complete.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.
- ➜ Fixed membership problem remains NP-complete.

Next steps

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.
- ➜ Fixed membership problem remains NP-complete.

Next steps

- ➜ Investigate closure properties of VSA.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.
- ➜ Fixed membership problem remains NP-complete.

Next steps

- ➜ Investigate closure properties of VSA.
- ➜ Investigate the time complexity of the emptiness problem.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.
- ➜ Fixed membership problem remains NP-complete.

Next steps

- ➜ Investigate closure properties of VSA.
- ➜ Investigate the time complexity of the emptiness problem.
- ➜ Investigate other decidability problems.

Summary

- ➜ We have introduced input-driven stack automata.
- ➜ Nondeterministic variants are more powerful than deterministic variants.
- ➜ Erasing variants are more powerful than non-erasing variants.
- ➜ Relaxing the motion inside the pushdown store is more powerful than input-driven motion.
- ➜ Fixed membership problem remains NP-complete.

Next steps

- ➜ Investigate closure properties of VSA.
- ➜ Investigate the time complexity of the emptiness problem.
- ➜ Investigate other decidability problems.
- ➜ Investigate the relations between nondeterministic variants.