Self-Verifying Pushdown Automata

Henning Fernau Martin Kutrib Matthias Wendlandt

Abteilung Informatik, Universität Trier

Institut für Informatik, Universität Giessen

Self-Verifying Pushdown Automata

Former Results

- Self-verifying finite automata have been introduced and studied by Ďuriš, Hromkovič, Rolim, Schnittger (1997) in connection with the study of Las Vegas automata.
- Descriptional complexity issues for self-verifying finite automata have been studied in Jirásková, Pighizzini (2011).
- The family of strongly context-free languages were studied by Ilie, Păun, Rozenberg, Salomaa (2000).
- Given a language such that also its complement belongs to the same family, the description of which of both is more economic (Jirásková 2005)?

Self-Verifying Pushdown Automata

Definition

- A self-verifying pushdown automata (SVPDA) is a nondeterministic pushdown automata so that each computation path can give one of the answers yes, no, or do not know.
- For every input word, at least one computation path must give either the answer yes or no.
- The answers given must not be contradictory.

Self-Verifying Pushdown Automata

Definition

- State set is partitioned into three subsets $Q = F_+ \cup F_- \cup F_0$.
- For each input word $w \in \Sigma^*$, the set Q_w of states reachable after processing the input entirely is defined as

$$\{\, q \in Q \mid (q_0, w, \bot) \vdash^* (q, \lambda, \gamma), \text{ for some } \gamma \in \Gamma^* \,\}.$$

- For an SVPDA it is required that, for each $w \in \Sigma^*$, $Q_w \cap F_+$ is empty if and only if $Q_w \cap F_-$ is nonempty.

Self-Verifying Pushdown Automata

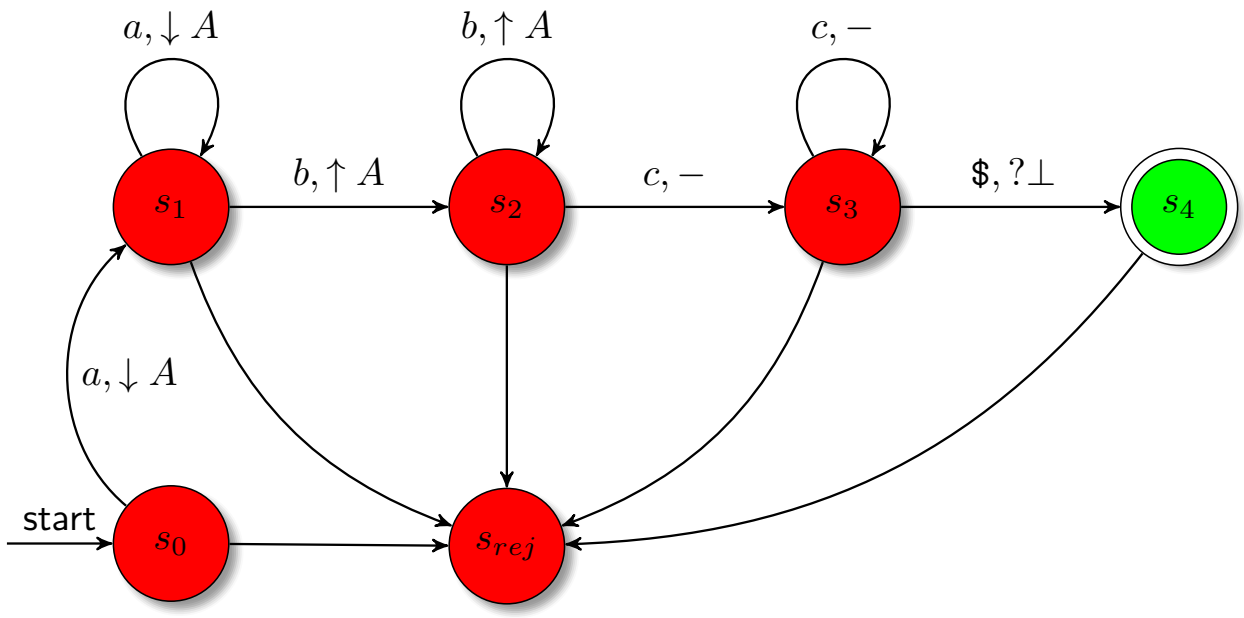
Example

- Is the language $\{ a^n b^n c^m \$ \mid m, n \geq 1 \} \cup \{ a^n b^m c^n \# \mid m, n \geq 1 \}$ accepted by an SVPDA?

Self-Verifying Pushdown Automata

Example

- Is the language $\{ a^n b^n c^m \$ \mid m, n \geq 1 \} \cup \{ a^n b^m c^n \# \mid m, n \geq 1 \}$ accepted by an SVPDA?

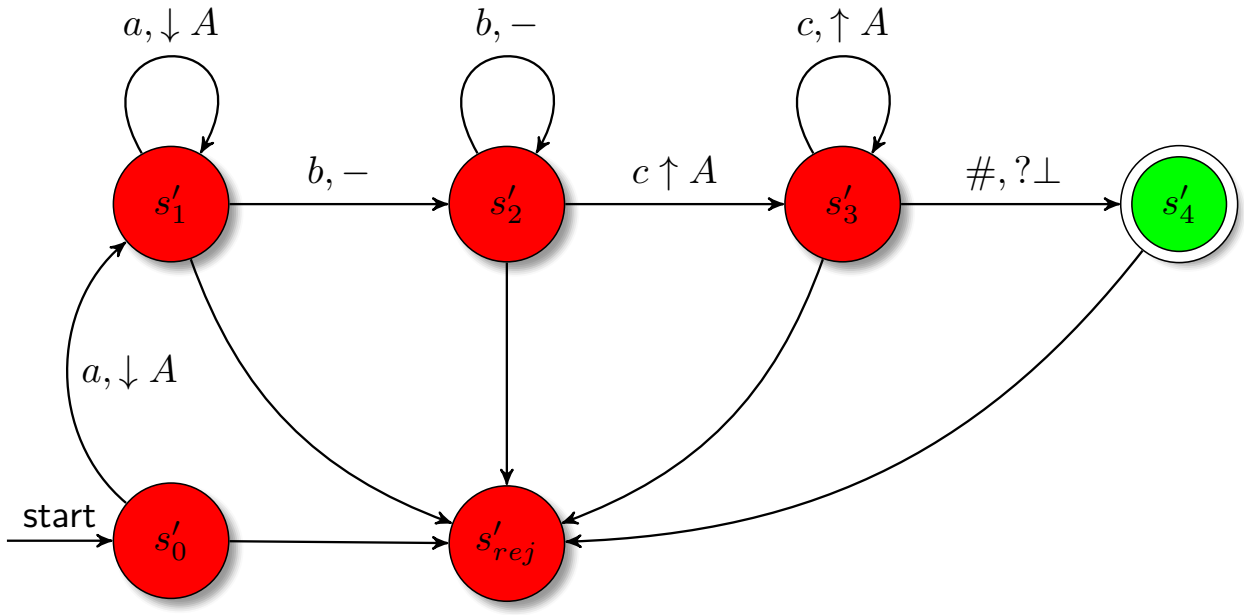


$w = aabbc\$$

Self-Verifying Pushdown Automata

Example

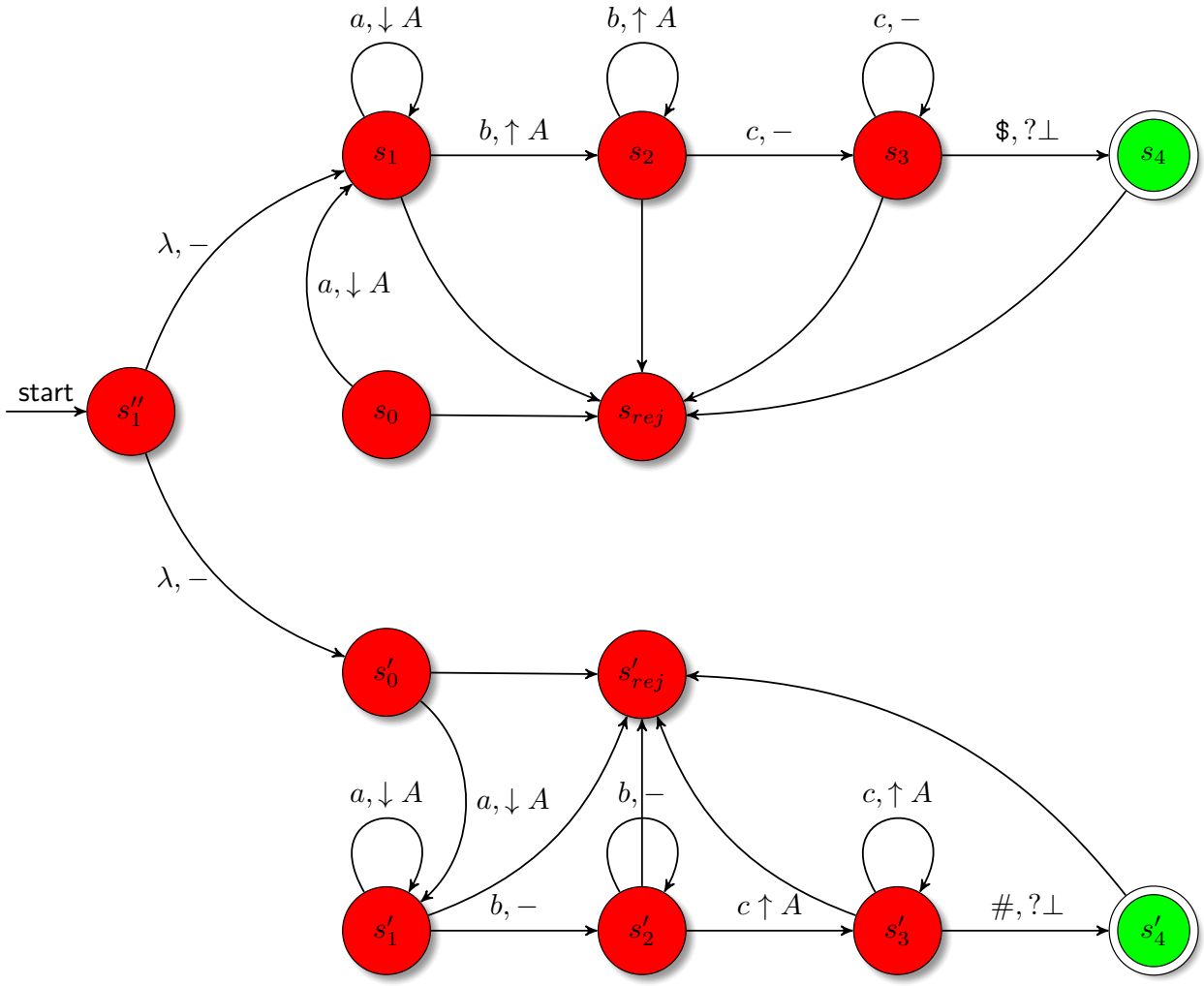
- Is the language $\{ a^n b^n c^m \$ \mid m, n \geq 1 \} \cup \{ a^n b^m c^n \# \mid m, n \geq 1 \}$ accepted by an SVPDA?



$w = aabcc\#$

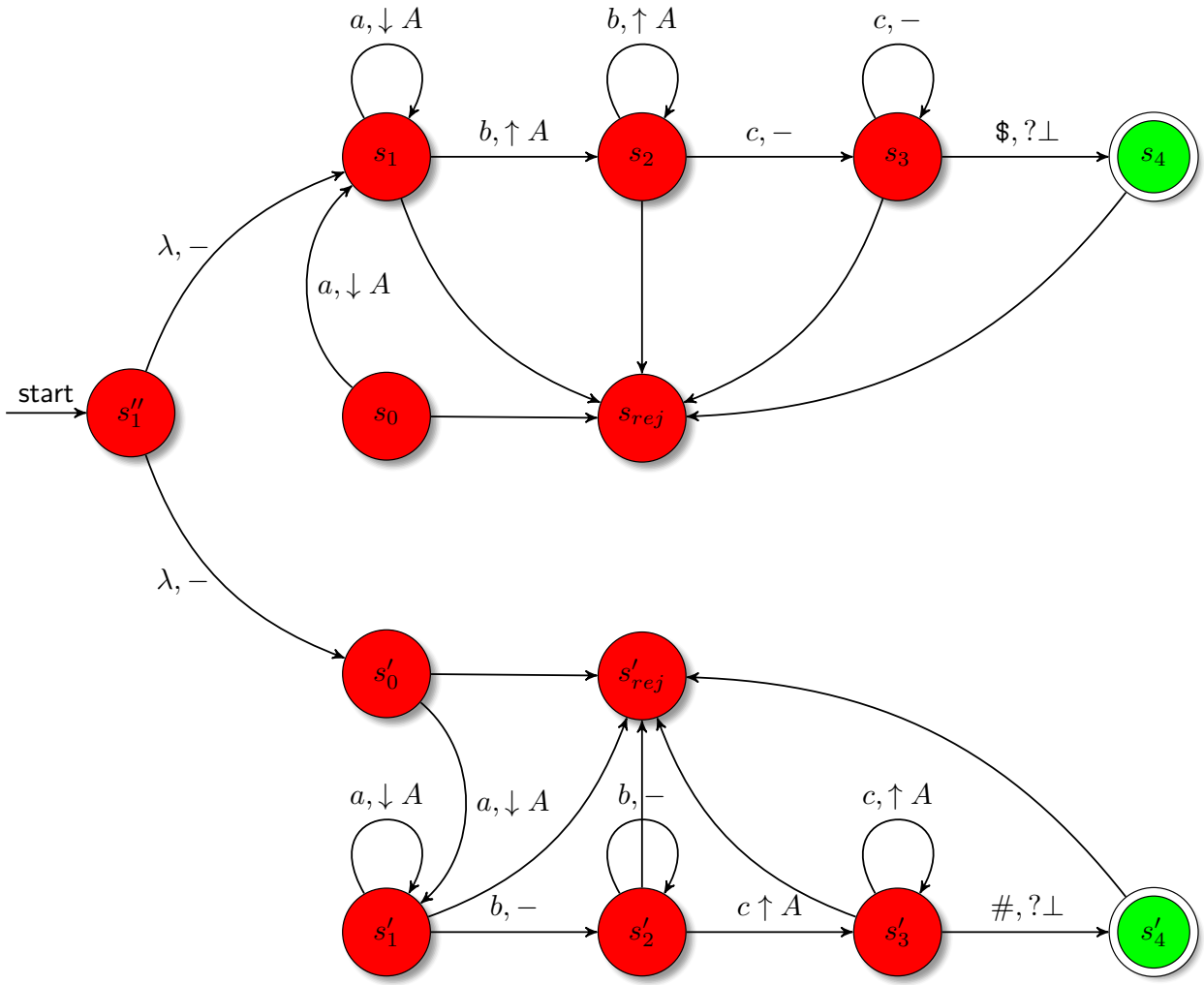
Self-Verifying Pushdown Automata

Solution?



Self-Verifying Pushdown Automata

Solution?



$w = aabbc\$$

Self-Verifying Pushdown Automata

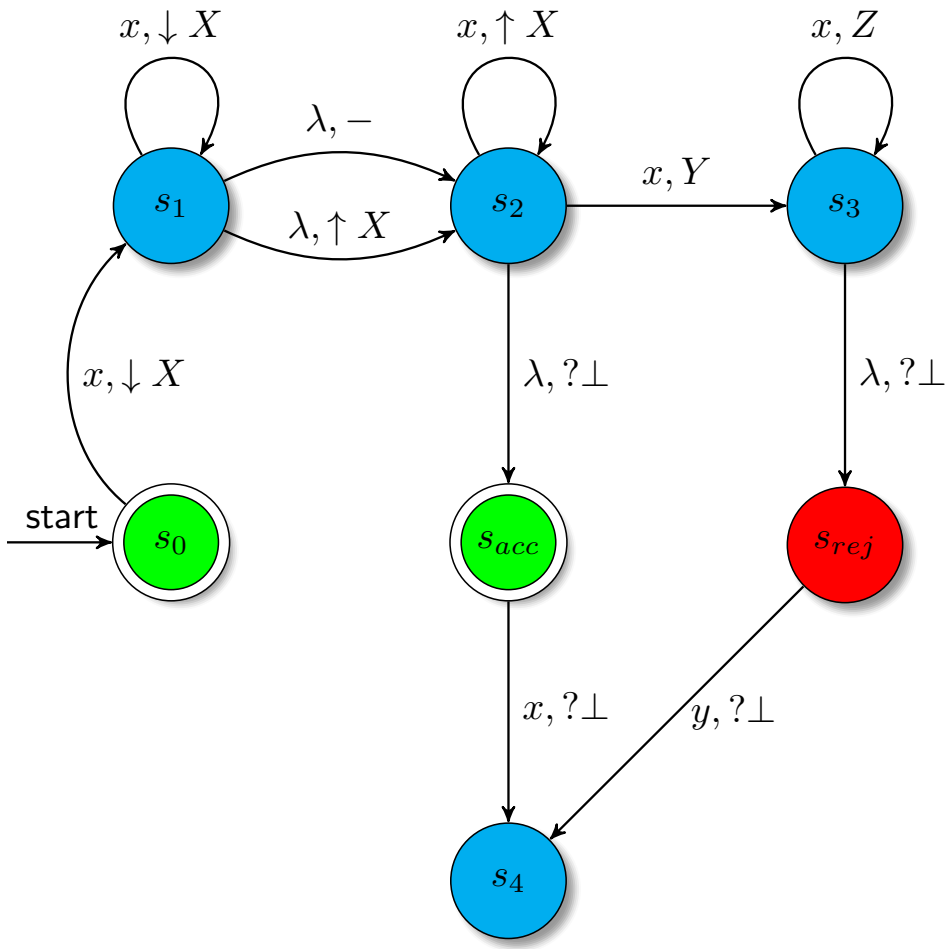
Example

The language $\{ w \mid w \in \{a, b\}^* \text{ and } w = w^R \}$ is accepted by an SVPDA.

Self-Verifying Pushdown Automata

Example

The language $\{ w \mid w \in \{a, b\}^* \text{ and } w = w^R \}$ is accepted by an SVPDA.



Self-Verifying Pushdown Automata – Computational Capacity

Self-Verifying Pushdown Automata

Strongly Context-Free Languages

Theorem

The families of strongly context-free languages and $\mathscr{L}(\mathrm{SVPDA})$ coincide.

Self-Verifying Pushdown Automata

Strongly Context-Free Languages

Theorem

The families of strongly context-free languages and $\mathscr{L}(\mathrm{SVPDA})$ coincide.

$$\mathscr{L}(\mathrm{DPDA}) \subset \mathscr{L}(\mathrm{SVPDA}) \subset \mathscr{L}(\mathrm{NPDA})$$

Self-Verifying Pushdown Automata

Strongly Context-Free Languages

Theorem

The families of strongly context-free languages and $\mathscr{L}(\mathrm{SVPDA})$ coincide.

$$\mathscr{L}(\mathrm{DPDA}) \subset \mathscr{L}(\mathrm{SVPDA}) \subset \mathscr{L}(\mathrm{NPDA})$$

Is the language $\{ a^n b^n c^m \$ \mid m, n \geq 0 \} \cup \{ a^n b^m c^n \# \mid m, n \geq 1 \}$ accepted by an SVPDA?

Self-Verifying Pushdown Automata – Basic Properties

Self-Verifying Pushdown Automata

Closure Properties

Family	$\overline{}$	$\cup$	$\cap$	$\cap_{REG}$	h_λ	h^{-1}	$\cdot$	$*$	R
$\mathscr{L}$(DPDA)	yes	no	no	yes	no	yes	no	no	no
$\mathscr{L}$(SVPDA)	yes	no	no	yes	no	yes	no	no	yes
$\mathscr{L}$(NPDA)	no	yes	no	yes	yes	yes	yes	yes	yes

Self-Verifying Pushdown Automata

Decidability Problems

	DPDA	SVPDA	NPDA
emptiness	$+$		$+$
finiteness	$+$		$+$
infiniteness	$+$		$+$
universality	$+$		$-$
inclusion	$-$		$-$
equivalence	$+$		$-$
regularity	$+$		$-$

- $+$ means *decidable*
- o means *semidecidable*
- $-$ means *not semidecidable*

Self-Verifying Pushdown Automata

Decidability Problems

	DPDA	SVPDA	NPDA
emptiness	$+$	$+$	$+$
finiteness	$+$	$+$	$+$
infiniteness	$+$	$+$	$+$
universality	$+$		$-$
inclusion	$-$	$-$	$-$
equivalence	$+$		$-$
regularity	$+$		$-$

- $+$ means *decidable*
- o means *semidecidable*
- $-$ means *not semidecidable*

Self-Verifying Pushdown Automata

Decidability Problems

	DPDA	SVPDA	NPDA
emptiness	$+$	$+$	$+$
finiteness	$+$	$+$	$+$
infiniteness	$+$	$+$	$+$
universality	$+$	$+$	$-$
inclusion	$-$	$-$	$-$
equivalence	$+$		$-$
regularity	$+$		$-$

- $+$ means *decidable*
- o means *semidecidable*
- $-$ means *not semidecidable*

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an $\mathrm{SVPDA}\ M$ and a regular language R. It is decidable whether $L(M) \subseteq R$ and whether $R \subseteq L(M)$.

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an $\mathrm{SVPDA}\ M$ and a regular language R. It is decidable whether $L(M) \subseteq R$ and whether $R \subseteq L(M)$.

Theorem

Regularity is semidecidable for self-verifying pushdown automata languages.

Self-Verifying Pushdown Automata

Decidability Problems

	DPDA	SVPDA	NPDA
emptiness	$+$	$+$	$+$
finiteness	$+$	$+$	$+$
infiniteness	$+$	$+$	$+$
universality	$+$	$+$	$-$
inclusion	$-$	$-$	$-$
equivalence	$+$		$-$
regularity	$+$	o	$-$

- $+$ means *decidable*
- o means *semidecidable*
- $-$ means *not semidecidable*

Self-Verifying Pushdown Automata

Decidability Problems

	DPDA	SVPDA	NPDA
emptiness	$+$	$+$	$+$
finiteness	$+$	$+$	$+$
infiniteness	$+$	$+$	$+$
universality	$+$	$+$	$-$
inclusion	$-$	$-$	$-$
equivalence	$+$	$?$	$-$
regularity	$+$	o	$-$

- $+$ means *decidable*
- o means *semidecidable*
- $-$ means *not semidecidable*

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an NPDA M and a partition of its state set into F_+, F_-, and F_0. It is undecidable whether or not M is an SVPDA.

- Let M_1 and M_2 be two arbitrary deterministic pushdown automata.
- We construct an NPDA M accepting $L(M_1) \cup L(M_2)$ that guesses whether to simulate M_1 or M_2.
- We define $F_+ = F_1$, $F_- = (Q_1 \setminus F_1) \cup F_2$, and $F_0 = Q_2 \setminus F_2$.

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an NPDA M and a partition of its state set into F_+, F_-, and F_0. It is undecidable whether or not M is an SVPDA.

- We define $F_+ = F_1$, $F_- = (Q_1 \setminus F_1) \cup F_2$, and $F_0 = Q_2 \setminus F_2$.

- $w \in L(M_1) \setminus L(M_2)$ – w is accepted by simulation of M_1.

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an NPDA M and a partition of its state set into F_+, F_-, and F_0. It is undecidable whether or not M is an SVPDA.

- We define $F_+ = F_1$, $F_- = (Q_1 \setminus F_1) \cup F_2$, and $F_0 = Q_2 \setminus F_2$.
- $w \in L(M_2) \setminus L(M_1)$ – then there is a rejecting path in M_2 as well as in M_1.

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an NPDA M and a partition of its state set into F_+, F_-, and F_0. It is undecidable whether or not M is an SVPDA.

- We define $F_+ = F_1$, $F_- = (Q_1 \setminus F_1) \cup F_2$, and $F_0 = Q_2 \setminus F_2$.
- $w \notin L(M_1) \cup L(M_2)$ – then there is a rejecting path in M_1 and a neutral path in M_2.

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an NPDA M and a partition of its state set into F_+, F_-, and F_0. It is undecidable whether or not M is an SVPDA.

- We define $F_+ = F_1$, $F_- = (Q_1 \setminus F_1) \cup F_2$, and $F_0 = Q_2 \setminus F_2$.
- $w \in L(M_1) \cap L(M_2)$ – then there is an accepting path in M_1 and a rejecting path in M_2.

Self-Verifying Pushdown Automata

Decidability

Theorem

Given an NPDA M and a partition of its state set into F_+, F_-, and F_0. It is undecidable whether or not M is an SVPDA.

- We define $F_+ = F_1$, $F_- = (Q_1 \setminus F_1) \cup F_2$, and $F_0 = Q_2 \setminus F_2$.
- $w \in L(M_1) \cap L(M_2)$ – then there is an accepting path in M_1 and a rejecting path in M_2.
- M is an SVPDA if and only if the intersection $L(M_1) \cap L(M_2)$ is empty.

Self-Verifying Pushdown Automata – Descriptional Complexity

Self-Verifying Pushdown Automata

Non-Recursive Trade-Offs

Theorem **[Holzer, K. 2010]**

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a procedure that decides $L(D)$,

Self-Verifying Pushdown Automata

Non-Recursive Trade-Offs

Theorem **[Holzer, K. 2010]**

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a procedure that decides $L(D)$, and let c_1 be a measure for $\mathcal{S}_1$ and c_2 be an sn-measure for $\mathcal{S}_2$.

Self-Verifying Pushdown Automata

Non-Recursive Trade-Offs

Theorem **[Holzer, K. 2010]**

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a procedure that decides $L(D)$, and let c_1 be a measure for $\mathcal{S}_1$ and c_2 be an sn-measure for $\mathcal{S}_2$.

If there exists a descriptional system $\mathcal{S}_3$ and a property P that is not semi-decidable for descriptors from $\mathcal{S}_3$,

Self-Verifying Pushdown Automata

Non-Recursive Trade-Offs

Theorem **[Holzer, K. 2010]**

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a procedure that decides $L(D)$, and let c_1 be a measure for $\mathcal{S}_1$ and c_2 be an sn-measure for $\mathcal{S}_2$.

If there exists a descriptional system $\mathcal{S}_3$ and a property P that is not semi-decidable for descriptors from $\mathcal{S}_3$, such that, given an arbitrary $D_3 \in \mathcal{S}_3$,

then the trade-off between S_1 and S_2 is non-recursive.

Self-Verifying Pushdown Automata

Non-Recursive Trade-Offs

Theorem **[Holzer, K. 2010]**

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a procedure that decides $L(D)$, and let c_1 be a measure for $\mathcal{S}_1$ and c_2 be an sn-measure for $\mathcal{S}_2$.

If there exists a descriptional system $\mathcal{S}_3$ and a property P that is not semi-decidable for descriptors from $\mathcal{S}_3$, such that, given an arbitrary $D_3 \in \mathcal{S}_3$,

(i) there exists an effective procedure to construct a descriptor D_1 in $\mathcal{S}_1$, and

then the trade-off between S_1 and S_2 is non-recursive.

Self-Verifying Pushdown Automata

Non-Recursive Trade-Offs

Theorem **[Holzer, K. 2010]**

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a procedure that decides $L(D)$, and let c_1 be a measure for $\mathcal{S}_1$ and c_2 be an sn-measure for $\mathcal{S}_2$.

If there exists a descriptional system $\mathcal{S}_3$ and a property P that is not semi-decidable for descriptors from $\mathcal{S}_3$, such that, given an arbitrary $D_3 \in \mathcal{S}_3$,

(i) there exists an effective procedure to construct a descriptor D_1 in $\mathcal{S}_1$, and

(ii) D_1 has an equivalent descriptor in $\mathcal{S}_2$ if and only if D_3 does not have property P,

then the trade-off between S_1 and S_2 is non-recursive.

Self-Verifying Pushdown Automata

Descriptional Complexity

- We use the family of deterministic one-tape Turing machines as descriptional system $\mathcal{S}_3$, and for the property P we take the property of accepting an infinite language.
  - $\mathrm{INVALC}(M)$ is a context-free language.
  - If M accepts an infinite language, $\mathrm{VALC}(M)$ is not even context free.
  - $\mathrm{VALC}(M)$ and $\mathrm{INVALC}(M)$ both are not accepted by any SVPDA.
  - If M accepts a finite language, then $\mathrm{VALC}(M)$ is finite and $\mathrm{INVALC}(M)$ is co-finite (regular) and thus accepted by some SVPDA.

Self-Verifying Pushdown Automata

Descriptional Complexity

Theorem

The trade-off between nondeterministic pushdown automata and self-verifying pushdown automata is non-recursive.

Self-Verifying Pushdown Automata

Descriptional Complexity

Theorem

The trade-off between self-verifying and deterministic pushdown automata is non-recursive.

Self-Verifying Pushdown Automata

Open and untouched Questions

- Is equivalence decidable?
- Is regularity decidable?
- Other models with the property of working self-verifying.

Thank you for your attention!