Two characterisation results of weighted multiple context-free grammars and their application to parsing

Statusvortrag

Tobias Denkinger

tobias.denkinger@tu-dresden.de

Institute of Theoretical Computer Science
Faculty of Computer Science
Technische Universität Dresden

2018-07-27

Outline

Outline

The k-best parsing problem

parsing problem

Input:

- a language device $\mathcal{M}$ (grammar or automaton)
- a word w

Output:

- an analysis of w in $\mathcal{M}$ (not unique)

The k-best parsing problem

k-best parsing problem [Jiménez and Marzal 2000]

Input:

- a $(\mathcal{A}, \odot, \mathbb{1}, \mathbb{0}, \trianglelefteq)$-weighted language device $(\mathcal{M}, \mathit{wt})$ (grammar or automaton)
- a number $k \in \mathbb{N}$
- a word w

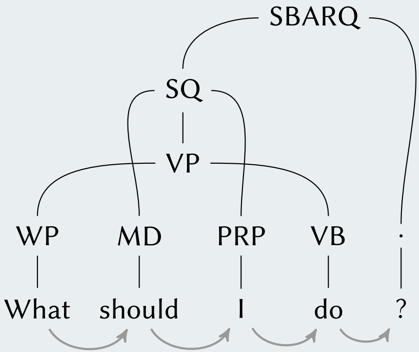
Output:

- a sequence of k best[1] analyses of w in $\mathcal{M}$ (not unique)

[1] w.r.t. *wt* and $\trianglelefteq$ (greater is better)

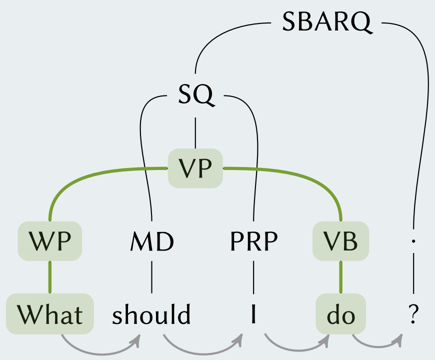
Discontinuous constituents and non-projective dependencies

Example [Evang and Kallmeyer 2011]



Discontinuous constituents and non-projective dependencies

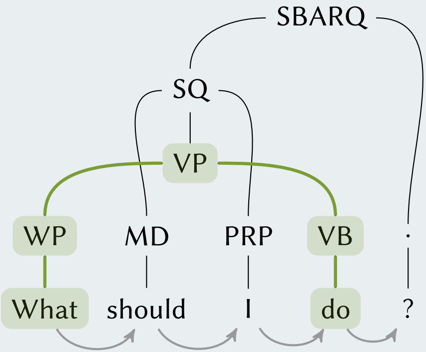
Example [Evang and Kallmeyer 2011]



gaps / crossing edges

Discontinuous constituents and non-projective dependencies

Example [Evang and Kallmeyer 2011]

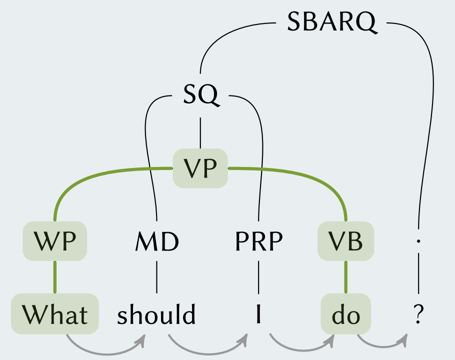


gaps / crossing edges

- not representable with CFGs

Discontinuous constituents and non-projective dependencies

Example [Evang and Kallmeyer 2011]

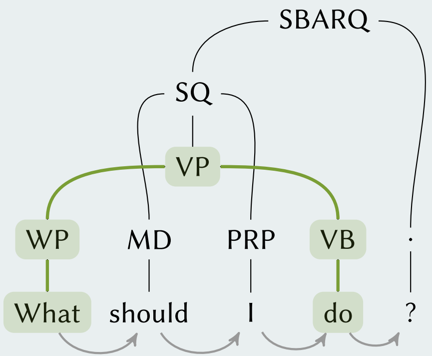


gaps / crossing edges

- not representable with CFGs
- occur in natural language tree banks

Discontinuous constituents and non-projective dependencies

Example [Evang and Kallmeyer 2011]



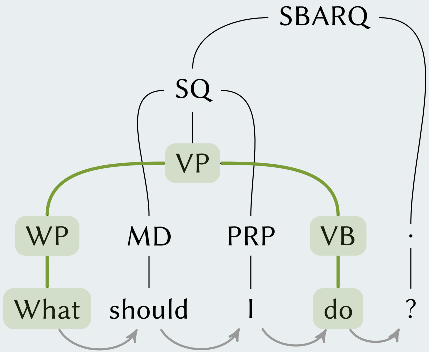
gaps / crossing edges

- not representable with CFGs
- occur in natural language tree banks

corpus	# of trees	cont.	discont.
[Maier and Søgaard 2008, Tab. 5]			
NeGra	$\approx 20\,000$	72.44%	27.56%
TIGER	$\approx 50\,000$	72.46%	27.54%
[Kuhlmann and Nivre 2006, Tab. 1]			
PDT	$\approx 73\,000$	76.85%	23.15%
DDT	$\approx 4\,400$	84.95%	15.05%

Discontinuous constituents and non-projective dependencies

Example [Evang and Kallmeyer 2011]



gaps / crossing edges

- not representable with CFGs
- occur in natural language tree banks

corpus	# of trees	cont.	discont.
[Maier and Søgaard 2008, Tab. 5]			
NeGra	≈ 20 000	72.44%	27.56%
TIGER	≈ 50 000	72.46%	27.54%
[Kuhlmann and Nivre 2006, Tab. 1]			
PDT	≈ 73 000	76.85%	23.15%
DDT	≈ 4 400	84.95%	15.05%

$\Longrightarrow$ use formalism more expressive than CFGs

Multiple context-free grammars

context-free grammars [Chomsky 1956]

$$A \to \mathrm{a}A\mathrm{b}B$$

composes strings

Multiple context-free grammars

context-free grammars [Chomsky 1956]

$$A \to \mathrm{a}A\mathrm{b}B$$

composes strings

$$A \to \underbrace{[(x, y) \mapsto \mathrm{a}x\mathrm{b}y]}_{\Sigma^* \times \Sigma^* \to \Sigma^*}(A, B)$$

Multiple context-free grammars

context-free grammars [Chomsky 1956]

$$A \to \mathrm{a}A\mathrm{b}B$$ composes strings

$$A \to [\quad \mathrm{a}\, x\, \mathrm{b}\, y \quad](A, B)$$

Multiple context-free grammars

context-free grammars [Chomsky 1956]

$$A \to \mathrm{a}A\mathrm{b}B$$

composes strings

$$A \to [\quad \mathrm{a}\, x\, \mathrm{b}\, y \quad](A, B)$$

***multiple* context-free grammars** [Seki, Matsumura, Fujii, and Kasami 1991]

$$A \to [\underbrace{((x_1, x_2), (y_1, y_2)) \mapsto (\mathrm{a}x_1y_2\mathrm{b}, y_1\mathrm{c}x_2)}_{(\Sigma^* \times \Sigma^*) \times (\Sigma^* \times \Sigma^*) \to (\Sigma^* \times \Sigma^*)}](A, B)$$

composes *tuples* of strings

Multiple context-free grammars

context-free grammars [Chomsky 1956]

$$A \to \mathrm{a}A\mathrm{b}B$$

composes strings

$$A \to [\ \mathrm{a}\,x\,\mathrm{b}\,y\ ](A, B)$$

***multiple* context-free grammars** [Seki, Matsumura, Fujii, and Kasami 1991]

$$A \to [\ \mathrm{a}x_1y_2\mathrm{b}\ ,\ y_1\mathrm{c}x_2\ ](A, B)$$

composes *tuples* of strings

Outline

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$.

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $mD \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $\quad L = h(mD \cap R)$.

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $\quad L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ and closure properties of $k\text{-MCF}(\mathcal{A})$

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $\quad L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ and closure properties of $k\text{-MCF}(\mathcal{A})$

(i) $\Rightarrow$ (ii) L

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $\quad L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ and closure properties of $k\text{-MCF}(\mathcal{A})$

(i) $\Rightarrow$ (ii) $\boxed{L}$

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

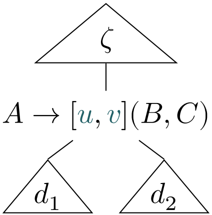
For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:

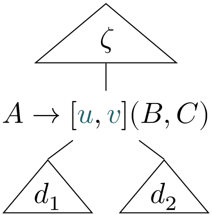


Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



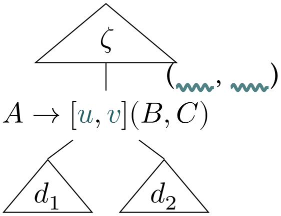
$\mu\colon R \to \mathcal{A}$

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



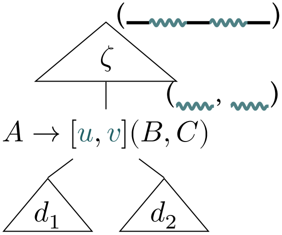
$\mu\colon R \to \mathcal{A}$

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



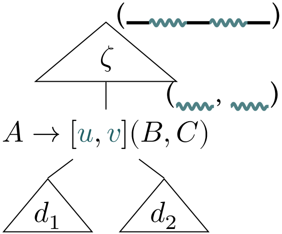
$\mu\colon R \to \mathcal{A}$

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

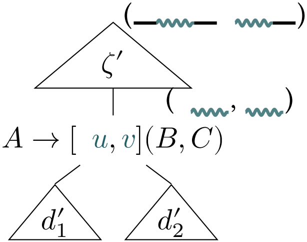
For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



$\mu\colon R \to \mathcal{A}$

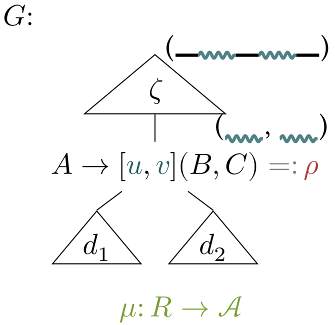
G' and h_1:

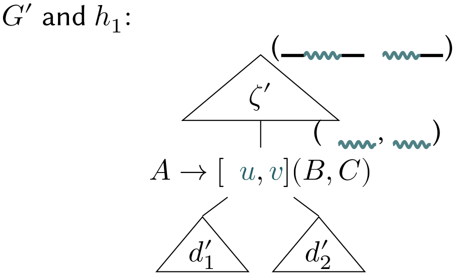


Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.



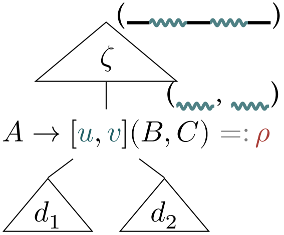


Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

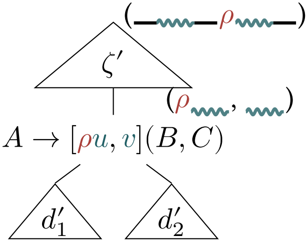
For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



$\mu\colon R \to \mathcal{A}$

G' and h_1:

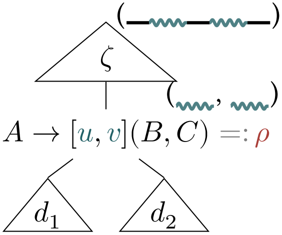


Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

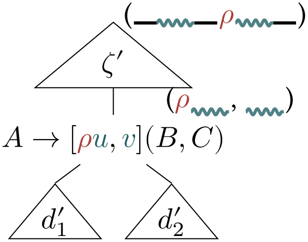
For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



$\mu\colon R \to \mathcal{A}$

G' and h_1:



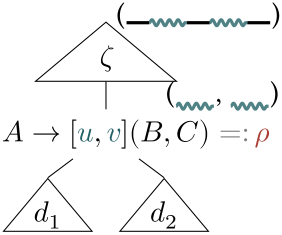
$$h_1(\delta) = \Bigg\{$$

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

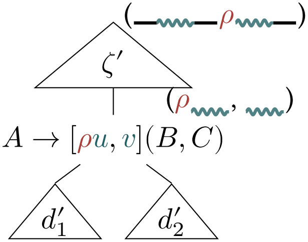
For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



$\mu\colon R \to \mathcal{A}$

G' and h_1:



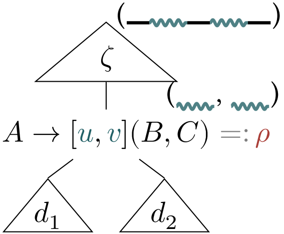
$$h_1(\delta) = \begin{cases} \mu(\rho).\varepsilon & \text{if } \delta = \rho, \rho \in R \end{cases}$$

Weight separation

Lemma [idea from Droste and Vogler 2013, Lem. 3]

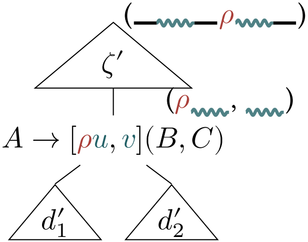
For every $\mathcal{A}$-weighted k-MCFL $L\colon \Sigma^* \to \mathcal{A}$ there are an $\mathcal{A}$-weighted α-homomorphism $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ and an unweighted k-MCFL $L' \subseteq (\Sigma \cup R)^*$ s.t. $L = h_1(L')$.

G:



$\mu\colon R \to \mathcal{A}$

G' and h_1:



$$h_1(\delta) = \begin{cases} \mu(\rho).\varepsilon & \text{if } \delta = \rho, \rho \in R \\ 1.\delta & \text{if } \delta \in \Sigma \end{cases}$$

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ and closure properties of $k\text{-MCF}(\mathcal{A})$

(i) $\Rightarrow$ (ii) $L = h_1(\ L'\ )$

- $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ is an $\mathcal{A}$-weighted α-homomorphism,
- $L' \subseteq (\Sigma \cup R)^*$ is k-multiple context-free,

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ and closure properties of $k\text{-MCF}(\mathcal{A})$

(i) $\Rightarrow$ (ii) $L = h_1(L')$

- $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ is an $\mathcal{A}$-weighted α-homomorphism,
- $L' \subseteq (\Sigma \cup R)^*$ is k-multiple context-free,

The unweighted Chomsky-Schützenberger theorem

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- a homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language $mD \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $L = h(mD \cap R)$.

The unweighted Chomsky-Schützenberger theorem

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- an α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $\quad L = h(\mathit{mD} \cap R)$.

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $\quad L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ and closure properties of $k\text{-MCF}(\mathcal{A})$

(i) $\Rightarrow$ (ii) $\quad L = h_1(\,L'\,) = h_1(h_2\,(R \cap \mathit{mD}))$

- $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ is an $\mathcal{A}$-weighted α-homomorphism,
- $L' \subseteq (\Sigma \cup R)^*$ is k-multiple context-free,
- $h_2\colon (\Delta \cup \bar{\Delta})^* \to (\Sigma \cup R)^*$ is an α-homomorphism

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language *mD* $\subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $L = h(\textit{mD} \cap R)$.

(ii) ⇒ (i) *mD* $\in k$-MCF and closure properties of k-MCF$(\mathcal{A})$

(i) ⇒ (ii) $L = h_1(\ L'\ ) = h_1(h_2(R \cap \textit{mD}))$

- $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ is an $\mathcal{A}$-weighted α-homomorphism,
- $L' \subseteq (\Sigma \cup R)^*$ is k-multiple context-free,
- $h_2\colon (\Delta \cup \bar{\Delta})^* \to (\Sigma \cup R)^*$ is an α-homomorphism

A Chomsky-Schützenberger characterisation for weighted MCFGs

Theorem [Denkinger 2015, Thm. 19]

Let $\mathcal{A}$ be a complete commutative strong bimonoid and $L\colon \Sigma^* \to \mathcal{A}$. For every $k \in \mathbb{N}$, the following are equivalent

(i) L is k-multiple context-free.

(ii) There are

- an $\mathcal{A}$-weighted α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^* \to \mathcal{A}$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

such that $\quad L = h(\mathit{mD} \cap R)$.

(ii) $\Rightarrow$ (i) $\mathit{mD} \in k\text{-MCF}$ $\quad$ and $\quad$ closure properties of $k\text{-MCF}(\mathcal{A})$

(i) $\Rightarrow$ (ii) $\qquad L \;= h_1(\; L' \;) = \; h_1(h_2\; (R \cap \mathit{mD})) = h(R \cap \mathit{mD})$

- $h_1\colon (\Sigma \cup R)^* \to \Sigma^* \to \mathcal{A}$ $\quad$ is an $\mathcal{A}$-weighted α-homomorphism,
- $L' \subseteq (\Sigma \cup R)^*$ $\quad$ is k-multiple context-free,
- $h_2\colon (\Delta \cup \bar{\Delta})^* \to (\Sigma \cup R)^*$ $\quad$ is an α-homomorphism

Outline

From the CS-theorem to CS-parsing

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- an α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language *mD* $\subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $\quad L = h(\mathit{mD} \cap R)$.

From the CS-theorem to CS-parsing

$$w \in \mathrm{L}(G) \qquad \text{(CS-theorem)}$$

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- an α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $\quad L = h(\mathit{mD} \cap R)$.

From the CS-theorem to CS-parsing

$$w \in \mathrm{L}(G) \iff w \in h(R \cap \mathit{mD}) \qquad \text{(CS-theorem)}$$

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- an α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $\quad L = h(\mathit{mD} \cap R)$.

From the CS-theorem to CS-parsing

$$\begin{aligned} w \in \mathrm{L}(G) &\iff w \in h(R \cap \mathit{mD}) && \text{(CS-theorem)} \\ &\iff \exists u \in R \cap \mathit{mD}\colon h(u) = w \end{aligned}$$

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- an α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language $\mathit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $\quad L = h(\mathit{mD} \cap R)$.

From the CS-theorem to CS-parsing

$$
\begin{aligned}
w \in \mathrm{L}(G) &\iff w \in h(R \cap \textit{mD}) \qquad \text{(CS-theorem)}\\
&\iff \exists u \in R \cap \textit{mD}\colon h(u) = w\\
&\iff \exists u \in R \cap h^{-1}(w)\colon u \in \textit{mD}
\end{aligned}
$$

Theorem [Yoshinaka, Kaji, and Seki 2010, Thm. 3]

Let $L \subseteq \Sigma^*$. For every $k \in \mathbb{N}$, t.f.a.e.

(i) L is k-multiple context-free.

(ii) There are an alphabet Δ,

- an α-homomorphism $h\colon (\Delta \cup \bar{\Delta})^* \to \Sigma^*$,
- a k-multiple Dyck language $\textit{mD} \subseteq (\Delta \cup \bar{\Delta})^*$, and
- a regular language $R \subseteq (\Delta \cup \bar{\Delta})^*$

s.t. $\quad L = h(\textit{mD} \cap R)$.

From the CS-theorem to CS-parsing

$$\begin{aligned} w \in \mathrm{L}(G) &\iff w \in h(R \cap \mathit{mD}) && \text{(CS-theorem)} \\ &\iff \exists u \in R \cap \mathit{mD}\colon h(u) = w \\ &\iff \exists u \in R \cap h^{-1}(w)\colon u \in \mathit{mD} \end{aligned}$$

From the CS-theorem to CS-parsing

$$\begin{aligned} w \in \mathrm{L}(G) &\iff w \in h(R \cap \boldsymbol{mD}) && \text{(CS-theorem)} \\ &\iff \exists u \in R \cap \boldsymbol{mD}\colon h(u) = w \\ &\iff \exists u \in R \cap h^{-1}(w)\colon u \in \boldsymbol{mD} \end{aligned}$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

From the CS-theorem to CS-parsing

$$
\begin{aligned}
w \in \mathrm{L}(G) &\iff w \in h(R \cap \boldsymbol{mD}) && \text{(CS-theorem)} \\
&\iff \exists u \in R \cap \boldsymbol{mD} \colon h(u) = w \\
&\iff \exists u \in R \cap h^{-1}(w) \colon u \in \boldsymbol{mD}
\end{aligned}
$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

k-best CS-parsing [Denkinger 2017b, Def. 5.21]

$$\mathrm{parse}_{G,\mathit{wt},k}(w)$$

From the CS-theorem to CS-parsing

$$\begin{aligned} w \in \mathrm{L}(G) &\iff w \in h(R \cap \mathit{mD}) && \text{(CS-theorem)} \\ &\iff \exists u \in R \cap \mathit{mD}\colon h(u) = w \\ &\iff \exists u \in R \cap h^{-1}(w)\colon u \in \mathit{mD} \end{aligned}$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

k-best CS-parsing [Denkinger 2017b, Def. 5.21]

$$\mathrm{parse}_{G,\mathit{wt},k}(w) = (\mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{toDeriv} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w))$$

From the CS-theorem to CS-parsing

$$\begin{aligned} w \in \mathrm{L}(G) &\iff w \in h(R \cap \mathit{mD}) && \text{(CS-theorem)} \\ &\iff \exists u \in R \cap \mathit{mD}\colon h(u) = w \\ &\iff \exists u \in R \cap h^{-1}(w)\colon u \in \mathit{mD} \end{aligned}$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

k-best CS-parsing [Denkinger 2017b, Def. 5.21]

$$\begin{aligned} \mathrm{parse}_{G,\mathit{wt},k}(w) &= (\mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{toDeriv} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w)) \\ &= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w)) \end{aligned}$$

From the CS-theorem to CS-parsing

$$\begin{aligned} w \in \mathrm{L}(G) &\iff w \in h(R \cap \mathit{mD}) && \text{(CS-theorem)} \\ &\iff \exists u \in R \cap \mathit{mD}\colon h(u) = w \\ &\iff \exists u \in R \cap h^{-1}(w)\colon u \in \mathit{mD} \end{aligned}$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

k-best CS-parsing [Denkinger 2017b, Def. 5.21]

$$\begin{aligned} \mathrm{parse}_{G,\mathit{wt},k}(w) &= (\mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{toDeriv} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w)) \\ &= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w)) \\ &= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap \mathit{mD}} \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq})(R \cap h^{-1}(w)) \end{aligned}$$

From the CS-theorem to CS-parsing

$$
\begin{aligned}
w \in \mathrm{L}(G) &\iff w \in h(R \cap \mathit{mD}) && \text{(CS-theorem)}\\
&\iff \exists u \in R \cap \mathit{mD}\colon h(u) = w\\
&\iff \exists u \in R \cap h^{-1}(w)\colon u \in \mathit{mD}
\end{aligned}
$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

k-best CS-parsing [Denkinger 2017b, Def. 5.21]

$$
\begin{aligned}
\mathrm{parse}_{G,\mathit{wt},k}(w) &= (\mathrm{take}_k \circ \mathrm{sort}_{\trianglelefteq}^{\mathit{wt}} \circ \mathrm{toDeriv} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w))\\
&= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{sort}_{\trianglelefteq}^{\mathit{wt}} \circ \mathrm{filter}_{\cap \mathit{mD}})(R \cap h^{-1}(w))\\
&= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap \mathit{mD}} \circ \mathrm{sort}_{\trianglelefteq}^{\mathit{wt}})(R \cap h^{-1}(w))\\
&= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap \mathit{mD}} \circ \mathrm{sort}_{\trianglelefteq})(R^{\mathit{wt}} \triangleright h^{-1}(w))
\end{aligned}
$$

From the CS-theorem to CS-parsing

$$
\begin{aligned}
w \in \mathrm{L}(G) &\iff w \in h(R \cap \boldsymbol{mD}) && \text{(CS-theorem)}\\
&\iff \exists u \in R \cap \boldsymbol{mD} \colon h(u) = w\\
&\iff \exists u \in R \cap h^{-1}(w) \colon u \in \boldsymbol{mD}
\end{aligned}
$$

Observation

Each $u \in R \cap D$ encodes a derivation of G.

k-best CS-parsing [Denkinger 2017b, Def. 5.21]

$$
\begin{aligned}
\mathrm{parse}_{G,\mathit{wt},k}(w) &= (\mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{toDeriv} \circ \mathrm{filter}_{\cap \boldsymbol{mD}})(R \cap h^{-1}(w))\\
&= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq} \circ \mathrm{filter}_{\cap \boldsymbol{mD}})(R \cap h^{-1}(w))\\
&= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap \boldsymbol{mD}} \circ \mathrm{sort}^{\mathit{wt}}_{\trianglelefteq})(R \cap h^{-1}(w))\\
&= (\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap \boldsymbol{mD}} \circ \underbrace{\mathrm{sort}_{\trianglelefteq})(R^{\mathit{wt}} \triangleright h^{-1}(w))}_{\text{enumerate from a weighted finite-state automaton}}
\end{aligned}
$$

Chomsky-Schützenberger parsing of weighted MCFGs

Theorem [Denkinger 2017b, Alg. 3 and Thm. 5.22]

Let (G, wt) be a *restricted* weighted MCFG over a *factorisable* $\trianglelefteq$-ordered monoid with zero where $\trianglelefteq$ is a partial order. Then

$$(\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap D} \circ \mathrm{sort}_{\trianglelefteq})(R^{\mathit{wt}} \triangleright h^{-1}(w))$$

solves the k-best parsing problem for (G, wt) and a word w.

Chomsky-Schützenberger parsing of weighted MCFGs

Theorem [Denkinger 2017b, Alg. 3 and Thm. 5.22]

Let (G, wt) be a *restricted* weighted MCFG over a *factorisable* $\trianglelefteq$-ordered monoid with zero where $\trianglelefteq$ is a partial order. Then

$$(\text{toDeriv} \circ \text{take}_k \circ \text{filter}_{\cap D} \circ \text{sort}_{\trianglelefteq})(R^{wt} \triangleright h^{-1}(w))$$

solves the k-best parsing problem for (G, wt) and a word w.

Conjecture

The restrictions are not problematic in practice.

Chomsky-Schützenberger parsing of weighted MCFGs

Theorem [Denkinger 2017b, Alg. 3 and Thm. 5.22]

Let (G, wt) be a *restricted* weighted MCFG over a *factorisable* $\trianglelefteq$-ordered monoid with zero where $\trianglelefteq$ is a partial order. Then

$$(\mathrm{toDeriv} \circ \mathrm{take}_k \circ \mathrm{filter}_{\cap D} \circ \mathrm{sort}_{\trianglelefteq})(R^{wt} \triangleright h^{-1}(w))$$

solves the k-best parsing problem for (G, wt) and a word w.

Conjecture

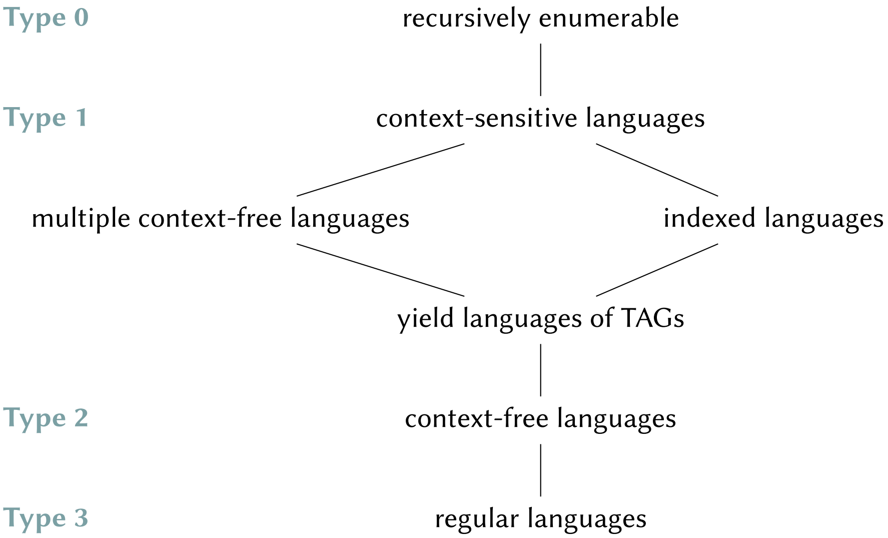
The restrictions are not problematic in practice.

- practical viability currently under investigation

Outline

1. Motivation and introduction
2. A Chomsky-Schützenberger characterisation for weighted MCFGs
3. Chomsky-Schützenberger parsing of weighted MCFGs
4. An automata characterisation of weighted MCFGs
5. Coarse-to-fine parsing of weighted automata with data storage

A set diagram of some language classes



A set diagram of some language classes

Type 0 recursively enumerable
↪Turing machines

Type 1 context-sensitive languages
↪linear bounded automata

multiple context-free languages

indexed languages
↪nested stack automata

yield languages of TAGs
↪embedded pushdown automata

Type 2 context-free languages
↪pushdown automata

Type 3 regular languages
↪finite state automata

A set diagram of some language classes

Type 0 recursively enumerable
↪Turing machines

Type 1 context-sensitive languages
↪linear bounded automata

multiple context-free languages
↪restricted tree-stack automata

indexed languages
↪nested stack automata

yield languages of TAGs
↪embedded pushdown automata

Type 2 context-free languages
↪pushdown automata

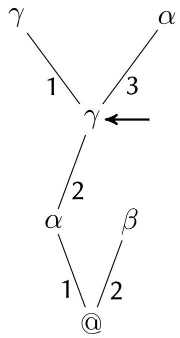
Type 3 regular languages
↪finite state automata

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ

example



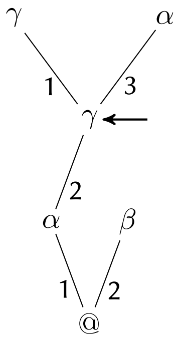
$\in \mathrm{TS}(\Gamma)$

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ

example

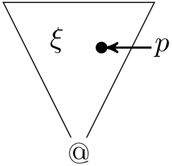


$\in \mathrm{TS}(\Gamma)$

The tree-stack idea from Villemonte de la Clergerie 2002a,b

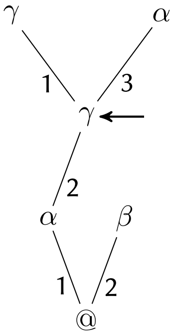
data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi \colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed



- @ exactly at the root

example

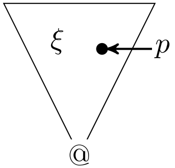


$\in \mathrm{TS}(\Gamma)$

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed

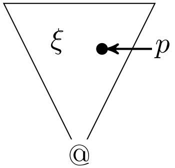


- @ exactly at the root

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed



- @ exactly at the root

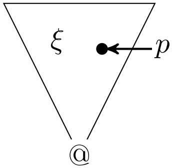
instructions (possibly partial)

$$\mathrm{eq}(\gamma)(\xi, p) = (\xi, p)$$
($\gamma \in \Gamma \cup \{@\}$, only if $\xi(p) = \gamma$)

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed



- @ exactly at the root

instructions (possibly partial)

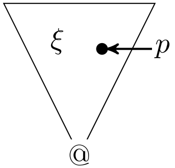
$\mathrm{eq}(\gamma)(\xi, p) = (\xi, p)$
($\gamma \in \Gamma \cup \{@\}$, only if $\xi(p) = \gamma$)

$\mathrm{set}(\gamma)$: set $\xi(p)$ to γ
($\gamma \in \Gamma$, only if $p \neq \varepsilon$)

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed



- @ exactly at the root

instructions (possibly partial)

$\mathrm{eq}(\gamma)(\xi, p) = \ (\xi, p)$
($\gamma \in \Gamma \cup \{@\}$, only if $\xi(p) = \gamma$)

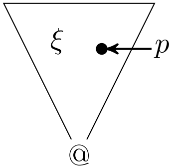
$\mathrm{set}(\gamma)$: set $\xi(p)$ to γ
($\gamma \in \Gamma$, only if $p \neq \varepsilon$)

up_i: move stack pointer to i-th child
(only if $\xi(pi)$ is defined)

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed



- @ exactly at the root

instructions (possibly partial)

$\mathrm{eq}(\gamma)(\xi, p) = (\xi, p)$
($\gamma \in \Gamma \cup \{@\}$, only if $\xi(p) = \gamma$)

$\mathrm{set}(\gamma)$: set $\xi(p)$ to γ
($\gamma \in \Gamma$, only if $p \neq \varepsilon$)

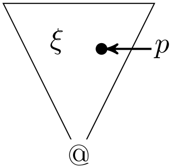
up_i: move stack pointer to i-th child
(only if $\xi(pi)$ is defined)

$\mathrm{push}_i(\gamma)$: push γ to the i-th child
(only if $\xi(pi)$ is undefined)

The tree-stack idea from Villemonte de la Clergerie 2002a,b

data storage $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $\xi\colon \mathbb{N}_+^* \dashrightarrow \Gamma \uplus \{@\}$
- stack pointer $p \in \mathbb{N}_+^*$ from the domain of ξ
- domain of ξ prefix-closed



- @ exactly at the root

instructions (possibly partial)

$\mathrm{eq}(\gamma)(\xi, p) = (\xi, p)$
  ($\gamma \in \Gamma \cup \{@\}$, only if $\xi(p) = \gamma$)

$\mathrm{set}(\gamma)$: set $\xi(p)$ to γ
  ($\gamma \in \Gamma$, only if $p \neq \varepsilon$)

up_i: move stack pointer to i-th child
  (only if $\xi(pi)$ is defined)

$\mathrm{push}_i(\gamma)$: push γ to the i-th child
  (only if $\xi(pi)$ is undefined)

down: move stack pointer to parent

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \qquad , 2)\\
\tau_2 &= (2, \mathrm{a}, \qquad , 2)\\
\tau_3 &= (2, \varepsilon, \qquad , 3)\\
\tau_4 &= (3, \varepsilon, \qquad , 3)\\
\tau_5 &= (3, \mathrm{b}, \qquad , 4)\\
\tau_6 &= (4, \mathrm{b}, \qquad , 4)\\
\tau_7 &= (4, \varepsilon, \qquad , 5)\\
\tau_8 &= (5, \varepsilon, \qquad , 5)\\
\tau_9 &= (5, \varepsilon, \qquad , 6)\\
\tau_{10} &= (6, \mathrm{c}, \qquad , 6)\\
\tau_{11} &= (6, \varepsilon, \qquad , 7)\\
\tau_{12} &= (7, \varepsilon, \qquad , 7)\\
\tau_{13} &= (7, \varepsilon, \qquad , 8)\\
\tau_{14} &= (8, \mathrm{d}, \qquad , 8)\\
\tau_{15} &= (8, \varepsilon, \qquad , 9)
\end{aligned}
$$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

instructions

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*), 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*), 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#), 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down}, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*), 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*), 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#), 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down}, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down}, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#), 9)
\end{aligned}
$$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3) \\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3) \\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4) \\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4) \\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5) \\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5) \\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6) \\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6) \\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7) \\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7) \\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8) \\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8) \\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}$$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*), 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*), 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#), 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down}, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*), 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*), 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#), 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down}, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down}, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#), 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 1 *stack:* @ $\leftarrow$

input tape:

a	a	b	c	c	d

run:

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
&\boxed{\tau_1 = (1, \mathrm{a}, \mathrm{push}_1(*), 2)}\\
&\tau_2 = (2, \mathrm{a}, \mathrm{push}_1(*), 2)\\
&\tau_3 = (2, \varepsilon, \mathrm{push}_1(\#), 3)\\
&\tau_4 = (3, \varepsilon, \mathrm{down}, 3)\\
&\tau_5 = (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*), 4)\\
&\tau_6 = (4, \mathrm{b}, \mathrm{push}_1(*), 4)\\
&\tau_7 = (4, \varepsilon, \mathrm{push}_1(\#), 5)\\
&\tau_8 = (5, \varepsilon, \mathrm{down}, 5)\\
&\tau_9 = (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1, 6)\\
&\tau_{10} = (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1, 6)\\
&\tau_{11} = (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 7)\\
&\tau_{12} = (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down}, 7)\\
&\tau_{13} = (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2, 8)\\
&\tau_{14} = (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1, 8)\\
&\tau_{15} = (8, \varepsilon, \mathrm{equals}(\#), 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

state: 1 *stack:* @ $\leftarrow$

input tape:

a	a	b	c	c	d

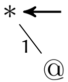
run:

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3) \\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3) \\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,; \mathrm{push}_2(*) &&, 4) \\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4) \\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5) \\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5) \\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_1 &&, 6) \\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 6) \\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,; \mathrm{down} &&, 7) \\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,; \mathrm{down} &&, 7) \\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_2 &&, 8) \\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 8) \\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 2 *stack:*



input tape:

a	a	b	c	c	d

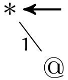
run: τ_1

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
&\boxed{\tau_2 = (2, \mathrm{a}, \mathrm{push}_1(*) \qquad\qquad , 2)} \\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3) \\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3) \\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4) \\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4) \\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5) \\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5) \\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6) \\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6) \\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7) \\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7) \\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8) \\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8) \\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

state: 2 *stack:*



input tape:

a	a	b	c	c	d

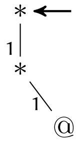
run: τ_1

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 2 *stack:*



input tape: | a | a | b | c | c | d |

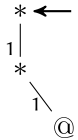
run: $\tau_1\tau_2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 2 *stack:*



input tape:

a	a	b	c	c	d

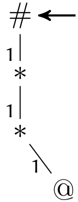
run: $\tau_1\tau_2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,; \mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,; \mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,; \mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

state: 3 *stack:*



input tape:

a	a	b	c	c	d

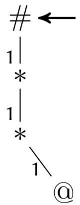
run: $\tau_1 \tau_2 \tau_3$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
&\boxed{\tau_4 = (3, \varepsilon, \mathrm{down} \quad, 3)}\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 3 *stack:*



input tape:

a	a	b	c	c	d

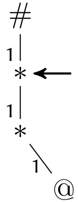
run: $\tau_1\tau_2\tau_3$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

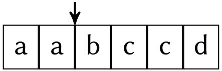
$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 3 *stack:*



input tape:



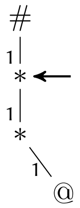
run: $\tau_1\tau_2\tau_3\tau_4$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*), 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*), 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#), 3)\\
&\boxed{\tau_4 = (3, \varepsilon, \mathrm{down}, 3)}\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*), 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*), 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#), 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down}, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down}, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#), 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

state: 3 *stack:*



input tape:

a	a	b	c	c	d

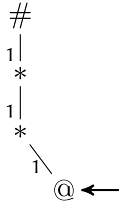
run: $\tau_1\tau_2\tau_3\tau_4$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 3 *stack:*



input tape:

a	a	b	c	c	d

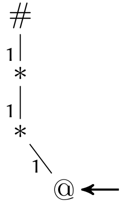
run: $\tau_1\tau_2\tau_3\tau_4^3$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
&\boxed{\tau_5 = (3, \mathrm{b}, \mathrm{bottom}\,; \mathrm{push}_2(*), 4)}\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,; \mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,; \mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 3 *stack:*



input tape:

a	a	b	c	c	d

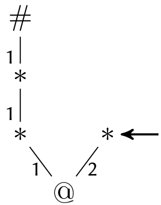
run: $\tau_1\tau_2\tau_3\tau_4^3$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,; \mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,; \mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,; \mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,; \mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,; \mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

state: 4 *stack:*



input tape:

a	a	b	c	c	d

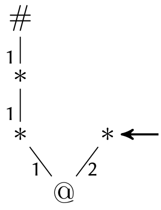
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

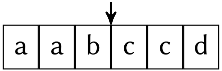
$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 4 *stack:*



input tape:



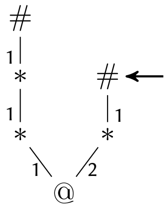
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 5 *stack:*



input tape:

a	a	b	c	c	d

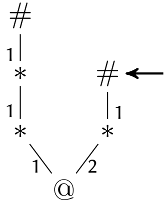
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) \quad , 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) \quad , 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) \quad , 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} \quad , 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*), 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) \quad , 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) \quad , 5)\\
&\boxed{\tau_8 = (5, \varepsilon, \mathrm{down} \quad , 5)}\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 \quad , 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 \quad , 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} \quad , 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 \quad , 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 \quad , 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) \quad , 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 5 *stack:*



input tape:

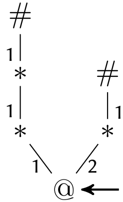
a	a	b	c	c	d

run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$



state: 5 *stack:*

input tape: | a | a | b | c | c | d |

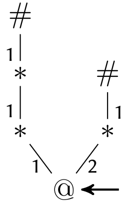
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 5 *stack:*



input tape:

a	a	b	c	c	d

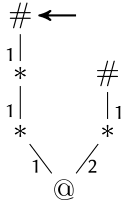
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 6 *stack:*



input tape:

a	a	b	c	c	d

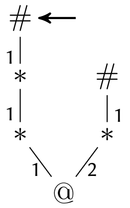
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2\tau_9\tau_{10}^2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 6 *stack:*



input tape:

a	a	b	c	c	d

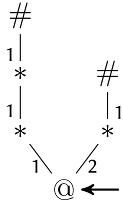
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2\tau_9\tau_{10}^2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 7 *stack:*



input tape:

a	a	b	c	c	d

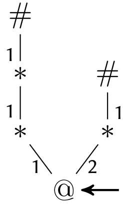
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2\tau_9\tau_{10}^2\tau_{11}\tau_{12}^2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3) \\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3) \\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4) \\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4) \\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5) \\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5) \\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6) \\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6) \\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7) \\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7) \\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8) \\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8) \\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^i \mathrm{d}^j \mid i, j \leq 1\}$

state: 7 *stack:*



input tape:

a	a	b	c	c	d

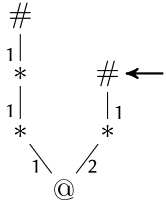
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2\tau_9\tau_{10}^2\tau_{11}\tau_{12}^2$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2) \\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3) \\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3) \\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4) \\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4) \\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5) \\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5) \\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6) \\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6) \\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7) \\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7) \\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8) \\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8) \\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 9 *stack:*



input tape:

a	a	b	c	c	d

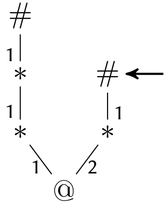
run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2\tau_9\tau_{10}^2\tau_{11}\tau_{12}^2\tau_{13}\tau_{14}\tau_{15}$

Tree-stack automata (TSA) as automata with storage, Scott (1967)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_2 &= (2, \mathrm{a}, \mathrm{push}_1(*) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{push}_1(\#) &&, 3)\\
\tau_4 &= (3, \varepsilon, \mathrm{down} &&, 3)\\
\tau_5 &= (3, \mathrm{b}, \mathrm{bottom}\,;\mathrm{push}_2(*) &&, 4)\\
\tau_6 &= (4, \mathrm{b}, \mathrm{push}_1(*) &&, 4)\\
\tau_7 &= (4, \varepsilon, \mathrm{push}_1(\#) &&, 5)\\
\tau_8 &= (5, \varepsilon, \mathrm{down} &&, 5)\\
\tau_9 &= (5, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 6)\\
\tau_{10} &= (6, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 6)\\
\tau_{11} &= (6, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 7)\\
\tau_{12} &= (7, \varepsilon, \mathrm{equals}(*)\,;\mathrm{down} &&, 7)\\
\tau_{13} &= (7, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_2 &&, 8)\\
\tau_{14} &= (8, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 8)\\
\tau_{15} &= (8, \varepsilon, \mathrm{equals}(\#) &&, 9)
\end{aligned}
$$

recognises $\{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^i\mathrm{d}^j \mid i, j \leq 1\}$

state: 9 *stack:*



input tape:

a	a	b	c	c	d

run: $\tau_1\tau_2\tau_3\tau_4^3\tau_5\tau_7\tau_8^2\tau_9\tau_{10}^2\tau_{11}\tau_{12}^2\tau_{13}\tau_{14}\tau_{15}$

2-restricted: enters each stack position at most 2 times *from below*

An automata characterisation of weighted MCFGs

Theorem [Denkinger 2016, Thm. 18]

$$k\text{-MCFL} = k\text{-TSL}_{\mathrm{r}}$$

Proof sketch: Show both set inclusions by construction.

---

k-MCFL: languages generated by MCFGs of fan-out at most k

k-TSL$_{\mathrm{r}}$: languages recognised by k-restricted tree stack automata

k-MCFL $\subseteq$ k-TSL$_{\mathrm{r}}$

Lemma [Denkinger 2016, $\approx$ Prop. 14]

$$k\text{-MCFL} \subseteq k\text{-TSL}_{\mathrm{r}}$$

k-MCFL $\subseteq$ k-TSL$_{\mathrm{r}}$

Lemma [Denkinger 2016, ≈ Prop. 14]

k-MCFL $\subseteq$ k-TSL$_{\mathrm{r}}$

Construction idea: return addresses

$$\rho = A \to [\quad \mathtt{a} \quad x_1 \quad , \quad \mathtt{c} \quad x_2 \quad](B)$$

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Lemma [Denkinger 2016, ≈ Prop. 14]

$$k\text{-MCFL} \subseteq k\text{-TSL}_\mathrm{r}$$

Construction idea:

return addresses

$$\rho = A \to [\underset{\langle\rho,1,0\rangle}{\bullet}\ \mathrm{a}\ \overset{\langle\rho,1,1\rangle}{\bullet}\ x_1\ \underset{\langle\rho,1,2\rangle}{\bullet}\ ,\ \overset{\langle\rho,2,0\rangle}{\bullet}\ \mathrm{c}\ \underset{\langle\rho,2,1\rangle}{\bullet}\ x_2\ \overset{\langle\rho,2,2\rangle}{\bullet}](B)$$

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Lemma [Denkinger 2016, ≈ Prop. 14]

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Construction idea:

return addresses

$$\rho = A \to [\, \underset{\uparrow\langle\rho,1,0\rangle}{\bullet}\ \mathsf{a}\ \overset{\langle\rho,1,1\rangle\downarrow}{\bullet}\ x_1\ \underset{\uparrow\langle\rho,1,2\rangle}{\bullet}\ ,\ \overset{\langle\rho,2,0\rangle\downarrow}{\bullet}\ \mathsf{c}\ \underset{\uparrow\langle\rho,2,1\rangle}{\bullet}\ x_2\ \overset{\langle\rho,2,2\rangle\downarrow}{\bullet}\,](B)$$

example transitions:

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Lemma [Denkinger 2016, ≈ Prop. 14]

$$k\text{-MCFL} \subseteq k\text{-TSL}_\mathrm{r}$$

Construction idea:

return addresses

$$\rho = A \to [\, \underset{\substack{\uparrow \\ \langle \rho, 1, 0\rangle}}{\bullet}\ \mathrm{a}\ \overset{\substack{\langle \rho, 1, 1\rangle \\ \downarrow}}{\bullet}\ x_1\ \underset{\substack{\uparrow \\ \langle \rho, 1, 2\rangle}}{\bullet}\ ,\ \overset{\substack{\langle \rho, 2, 0\rangle \\ \downarrow}}{\bullet}\ \mathrm{c}\ \underset{\substack{\uparrow \\ \langle \rho, 2, 1\rangle}}{\bullet}\ x_2\ \overset{\substack{\langle \rho, 2, 2\rangle \\ \downarrow}}{\bullet}\,](B)$$

example transitions:

read: $(\langle \rho, 1, 0\rangle,\ a,\ \mathrm{id},\ \langle \rho, 1, 1\rangle)$

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Lemma [Denkinger 2016, ≈ Prop. 14]

$$k\text{-MCFL} \subseteq k\text{-TSL}_\mathrm{r}$$

Construction idea:

return addresses

$$\rho = A \to [\underset{\uparrow \langle\rho,1,0\rangle}{\bullet}\ \mathrm{a}\ \overset{\langle\rho,1,1\rangle \downarrow}{\bullet}\ x_1\ \underset{\uparrow \langle\rho,1,2\rangle}{\bullet}\ ,\ \overset{\langle\rho,2,0\rangle \downarrow}{\bullet}\ \mathrm{c}\ \underset{\uparrow \langle\rho,2,1\rangle}{\bullet}\ x_2\ \overset{\langle\rho,2,2\rangle \downarrow}{\bullet}\ ](B)$$

example transitions:

(ρ' has lhs B and $\bar\rho$ has A on rhs)

read: $(\langle\rho,1,0\rangle, a, \mathrm{id}, \langle\rho,1,1\rangle)$

call: $(\langle\rho,1,1\rangle, \varepsilon, \mathrm{push}_1(\langle\rho,1,2\rangle), \langle\rho',1,0\rangle)$

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Lemma [Denkinger 2016, ≈ Prop. 14]

$$k\text{-MCFL} \subseteq k\text{-TSL}_\mathrm{r}$$

Construction idea: return addresses

$$\rho = A \to [\underset{\uparrow \langle \rho,1,0\rangle}{\bullet}\ \mathrm{a}\ \overset{\langle \rho,1,1\rangle \downarrow}{\bullet}\ x_1\ \underset{\uparrow \langle \rho,1,2\rangle}{\bullet}\ ,\ \overset{\langle \rho,2,0\rangle \downarrow}{\bullet}\ \mathrm{c}\ \underset{\uparrow \langle \rho,2,1\rangle}{\bullet}\ x_2\ \overset{\langle \rho,2,2\rangle \downarrow}{\bullet}\ ](B)$$

example transitions: (ρ' has lhs B and $\bar{\rho}$ has A on rhs)

read: $(\langle \rho, 1, 0\rangle, a, \mathrm{id}, \langle \rho, 1, 1\rangle)$

call: $(\langle \rho, 1, 1\rangle, \varepsilon, \mathrm{push}_1(\langle \rho, 1, 2\rangle), \langle \rho', 1, 0\rangle)$

return: $(\langle \rho, 1, 2\rangle, \varepsilon, \mathrm{equals}(\langle \bar{\rho}, i, j\rangle)\,;\mathrm{set}(\rho)\,;\mathrm{down}, \langle \bar{\rho}, i, j\rangle)$

k-MCFL $\subseteq$ k-TSL$_\mathrm{r}$

Lemma [Denkinger 2016, ≈ Prop. 14]

$$k\text{-MCFL} \subseteq k\text{-TSL}_\mathrm{r}$$

Construction idea:

return addresses

$$\rho = A \to [\underset{\uparrow \langle \rho,1,0\rangle}{\bullet}\ \mathrm{a}\ \overset{\langle \rho,1,1\rangle \downarrow}{\bullet}\ x_1\ \underset{\uparrow \langle \rho,1,2\rangle}{\bullet}\ ,\ \overset{\langle \rho,2,0\rangle \downarrow}{\bullet}\ \mathrm{c}\ \underset{\uparrow \langle \rho,2,1\rangle}{\bullet}\ x_2\ \overset{\langle \rho,2,2\rangle \downarrow}{\bullet}\ ](B)$$

example transitions:

(ρ' has lhs B and $\bar{\rho}$ has A on rhs)

read: $(\langle \rho, 1, 0\rangle,\ a,\ \mathrm{id},\ \langle \rho, 1, 1\rangle)$

call: $(\langle \rho, 1, 1\rangle,\ \varepsilon,\ \mathrm{push}_1(\langle \rho, 1, 2\rangle),\ \langle \rho', 1, 0\rangle)$

return: $(\langle \rho, 1, 2\rangle, \varepsilon, \mathrm{equals}(\langle \bar{\rho}, i, j\rangle)\,;\mathrm{set}(\rho)\,;\mathrm{down}, \langle \bar{\rho}, i, j\rangle)$

resume: $(\langle \rho, 2, 1\rangle, \varepsilon, \mathrm{up}_1\,;\mathrm{equals}(\rho')\,;\mathrm{set}(\langle \rho, 2, 2\rangle), \langle \rho', 2, 0\rangle)$

$k\text{-TSL}_{\mathrm{r}} \subseteq k\text{-MCFL}$

Lemma [Denkinger 2016, ≈ Prop. 17]

$$k\text{-TSL}_{\mathrm{r}} \subseteq k\text{-MCFL}$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL

Lemma [Denkinger 2016, $\approx$ Prop. 17]

$$k\text{-TSL}_\mathrm{r} \subseteq k\text{-MCFL}$$

Proof idea:

(1) construct an MCFG that generates the runs

(2) use closure of MCFGs under homomorphisms

[Seki, Matsumura, Fujii, and Kasami 1991, Thm. 3.9]

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL

Lemma [Denkinger 2016, $\approx$ Prop. 17]

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL

Proof idea:

(1) construct an MCFG that generates the runs

$$\langle \underbrace{q_1, q_1', \ldots, q_m, q_m'}_{\in Q^{2m}}; \underbrace{\gamma_0, \ldots, \gamma_m}_{\in \Gamma^{m+1}} \rangle \Longrightarrow^* (\theta_1, \ldots, \theta_m)$$

if and only if

- $\theta_1, \ldots, \theta_m$ all return to the stack position they started from and never go below it
- θ_i starts with state q_i and stack symbol γ_{i-1} and ends with q_i' and γ_i (for $1 \leq i \leq m$)

(2) use closure of MCFGs under homomorphisms
[Seki, Matsumura, Fujii, and Kasami 1991, Thm. 3.9]

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) , 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) , 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} , 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 , 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 , 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} , 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} , 5)
\end{aligned}$$

run for $a^2b^2c^2d^2$ and the stack:

111

11

1

ε $(1, @)$

k-TSL$_r$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:

111

11

1 $(1, *)$

$\tau_1 \uparrow$

ε $(1, @)$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}$$

run for $a^2b^2c^2d^2$ and the stack:

$$\begin{array}{rl}
111 & \\
11 & (1, *)\\
 & \uparrow \tau_1\\
1 & (1, *)\\
 & \uparrow \tau_1\\
\varepsilon & (1, @)
\end{array}$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:

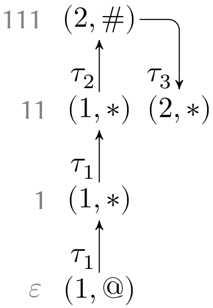
$$
\begin{tikzcd}
111 & (2, \#) \\
11 & (1, *) \arrow[u, "\tau_2"] \\
1 & (1, *) \arrow[u, "\tau_1"] \\
\varepsilon & (1, @) \arrow[u, "\tau_1"]
\end{tikzcd}
$$

k-TSL$_r$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:

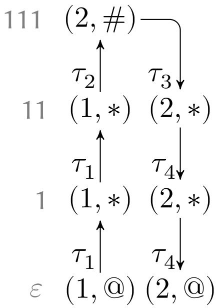


k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}$$

run for $a^2b^2c^2d^2$ and the stack:

$$\begin{tikzcd}
111 & (2,\#) \arrow[dr, bend left, "\tau_3"] & & (3,\#) \\
11 & (1,*) \arrow[u, "\tau_2"] & (2,*) \arrow[d, "\tau_4"] & (3,*) \arrow[u, "\tau_6"] \\
1 & (1,*) \arrow[u, "\tau_1"] & (2,*) \arrow[d, "\tau_4"] & (3,*) \arrow[u, "\tau_6"] \\
\varepsilon & (1,@) \arrow[u, "\tau_1"] & (2,@) \arrow[ur, bend right, "\tau_5"'] &
\end{tikzcd}$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1) \\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2) \\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2) \\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2) \\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3) \\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3) \\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4) \\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4) \\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:

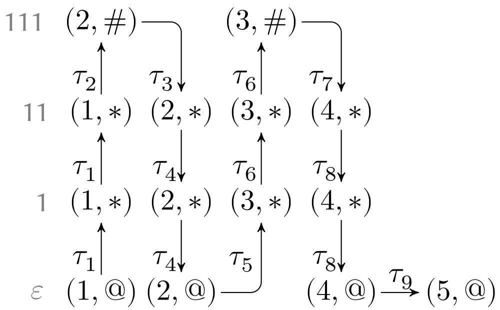
$$
\begin{tikzcd}
111 & (2,\#) \arrow[dr, "\tau_3"] & & (3,\#) \arrow[dr, "\tau_7"] & \\
11 & (1,*) \arrow[u, "\tau_2"] & (2,*) \arrow[d, "\tau_4"] & (3,*) \arrow[u, "\tau_6"] & (4,*) \arrow[d, "\tau_8"] \\
1 & (1,*) \arrow[u, "\tau_1"] & (2,*) \arrow[d, "\tau_4"] & (3,*) \arrow[u, "\tau_6"] & (4,*) \arrow[d, "\tau_8"] \\
\varepsilon & (1,@) \arrow[u, "\tau_1"] & (2,@) \arrow[ur, "\tau_5"] & & (4,@)
\end{tikzcd}
$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:

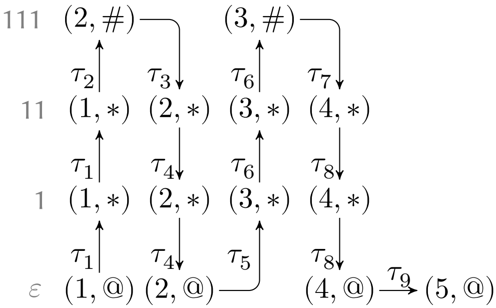


k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



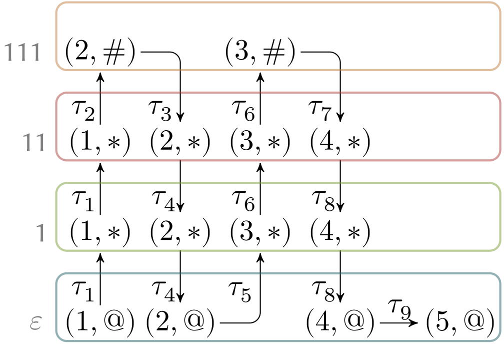
principle of crossing sequences

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1) \\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2) \\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#) ; \mathrm{down} &&, 2) \\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*) ; \mathrm{down} &&, 2) \\
\tau_5 &= (2, \varepsilon, \mathrm{bottom} ; \mathrm{up}_1 &&, 3) \\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*) ; \mathrm{up}_1 &&, 3) \\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#) ; \mathrm{down} &&, 4) \\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*) ; \mathrm{down} &&, 4) \\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



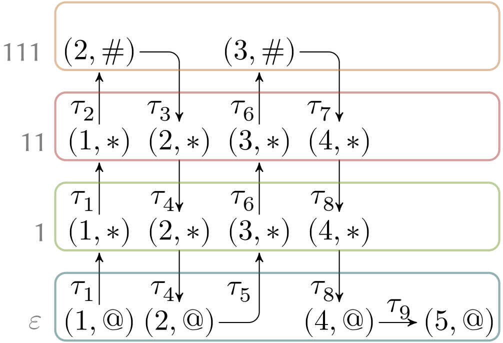
principle of crossing sequences

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}$$

run for $a^2b^2c^2d^2$ and the stack:



some rules:

principle of crossing sequences

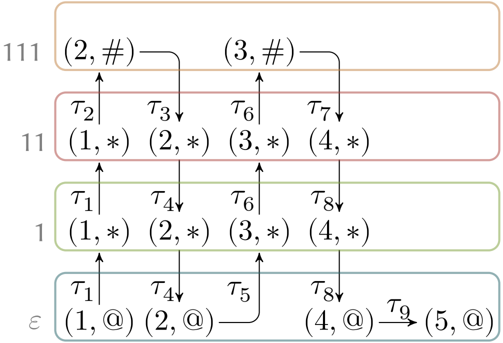
$$\langle 1, 5; @, @\rangle \to [\tau_1 x_1 \tau_4 \tau_5 x_2 \tau_8 \tau_9](\langle 1, 2, 3, 4; *, *, *\rangle)$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



some rules:

principle of crossing sequences

$$\langle 1,2,3,4;*,*,*\rangle \to [\tau_1 x_1 \tau_4, \tau_6 x_2 \tau_8](\langle 1,2,3,4;*,*,*\rangle)$$

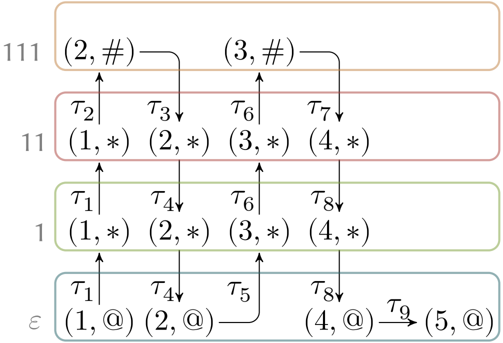
$$\langle 1,5;@,@\rangle \to [\tau_1 x_1 \tau_4 \tau_5 x_2 \tau_8 \tau_9](\langle 1,2,3,4;*,*,*\rangle)$$

k-TSL$_r$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}&&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down}&&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down}&&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down}&&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



some rules:

principle of crossing sequences

$$\langle 1,2,3,4; *,*,* \rangle \to [\tau_2 x_1 \tau_3, \tau_6 x_2 \tau_7](\langle 2,2,3,3; \#,\#,\# \rangle)$$

$$\langle 1,2,3,4; *,*,* \rangle \to [\tau_1 x_1 \tau_4, \tau_6 x_2 \tau_8](\langle 1,2,3,4; *,*,* \rangle)$$

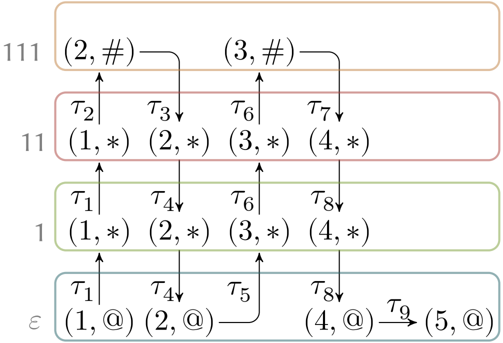
$$\langle 1,5; @,@ \rangle \to [\tau_1 x_1 \tau_4 \tau_5 x_2 \tau_8 \tau_9](\langle 1,2,3,4; *,*,* \rangle)$$

k-TSL$_r$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



some rules:

principle of crossing sequences

$$\langle 2, 2, 3, 3; \#, \#, \# \rangle \to [\varepsilon, \varepsilon]()$$

$$\langle 1, 2, 3, 4; *, *, * \rangle \to [\tau_2 x_1 \tau_3, \tau_6 x_2 \tau_7](\langle 2, 2, 3, 3; \#, \#, \# \rangle)$$

$$\langle 1, 2, 3, 4; *, *, * \rangle \to [\tau_1 x_1 \tau_4, \tau_6 x_2 \tau_8](\langle 1, 2, 3, 4; *, *, * \rangle)$$

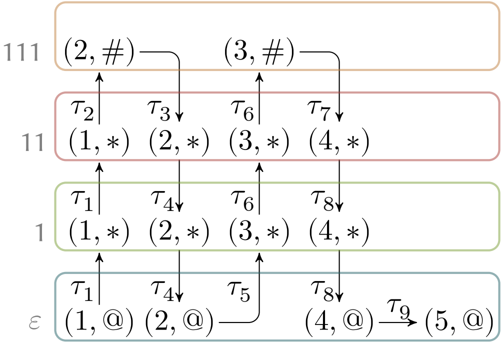
$$\langle 1, 5; @, @ \rangle \to [\tau_1 x_1 \tau_4 \tau_5 x_2 \tau_8 \tau_9](\langle 1, 2, 3, 4; *, *, * \rangle)$$

k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (monadic example)

transitions: (in one-move normal form)

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{equals}(*)\,;\mathrm{down} &&, 2)\\
\tau_5 &= (2, \varepsilon, \mathrm{bottom}\,;\mathrm{up}_1 &&, 3)\\
\tau_6 &= (3, \mathrm{c}, \mathrm{equals}(*)\,;\mathrm{up}_1 &&, 3)\\
\tau_7 &= (3, \varepsilon, \mathrm{equals}(\#)\,;\mathrm{down} &&, 4)\\
\tau_8 &= (4, \mathrm{d}, \mathrm{equals}(*)\,;\mathrm{down} &&, 4)\\
\tau_9 &= (4, \varepsilon, \mathrm{bottom} &&, 5)
\end{aligned}
$$

run for $a^2b^2c^2d^2$ and the stack:



some rules:

principle of crossing sequences

$$
\begin{aligned}
\langle 2,2,3,3;\#,\#,\#\rangle &\to [\varepsilon,\varepsilon]()\\
\langle 1,2,3,4;*,*,*\rangle &\to [\ x_1\ ,\ \mathrm{c}\,x_2\ ](\langle 2,2,3,3;\#,\#,\#\rangle)\\
\langle 1,2,3,4;*,*,*\rangle &\to [\,\mathrm{a}\,x_1\,\mathrm{b}\,,\ \mathrm{c}\,x_2\,\mathrm{d}\,](\langle 1,2,3,4;*,*,*\rangle)\\
\langle 1,5;@,@\rangle &\to [\,\mathrm{a}\,x_1\,\mathrm{b}\quad x_2\,\mathrm{d}\ ](\langle 1,2,3,4;*,*,*\rangle)
\end{aligned}
$$

Outline

Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

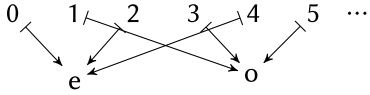
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



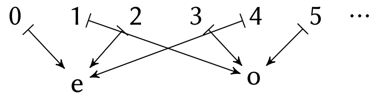
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$

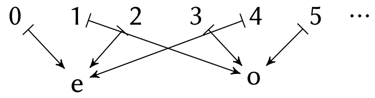
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0))$

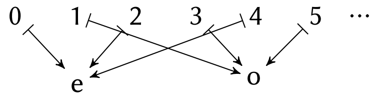
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$

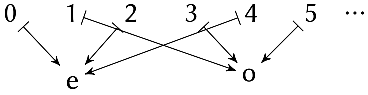
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $\quad A_{\mathrm{eo}}(\mathrm{inc}) = \{ \qquad \}$

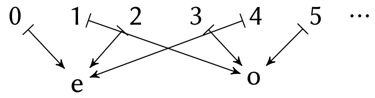
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $\qquad A_{\mathrm{eo}}(\mathrm{inc}) = \{ \qquad \}$

$$\begin{tikzcd} \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\ {} \end{tikzcd}$$

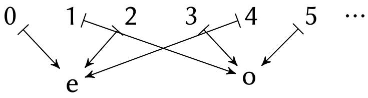
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e},\mathrm{e})\}}_{(=\mathrm{e})}$ $\qquad A_{\mathrm{eo}}(\mathrm{inc}) = \{ \qquad\qquad \}$

$$\begin{tikzcd}
\{0,2,4,\ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
& {}
\end{tikzcd}$$

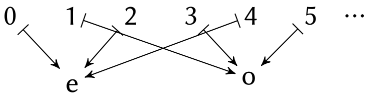
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $\quad A_{\mathrm{eo}}(\mathrm{inc}) = \{ \qquad \}$

$$\begin{tikzcd}
\{0, 2, 4, \ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{1, 3, 5, \ldots\} &
\end{tikzcd}$$

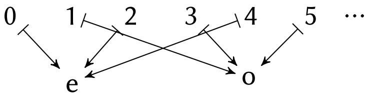
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $\quad A_{\mathrm{eo}}(\mathrm{inc}) = \{(\mathrm{e}, \mathrm{o}) \quad \}$

$$\begin{tikzcd}
\{0, 2, 4, \dots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{1, 3, 5, \dots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{o}\}
\end{tikzcd}$$

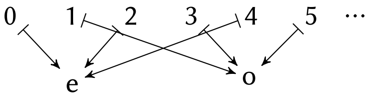
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $\qquad A_{\mathrm{eo}}(\mathrm{inc}) = \{(\mathrm{e}, \mathrm{o}), (\mathrm{o}, \mathrm{e})\}$

$$\begin{tikzcd}
\{0, 2, 4, \dots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{1, 3, 5, \dots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{o}\}
\end{tikzcd}
\qquad
\begin{tikzcd}
\{1, 3, 5, \dots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{o}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{2, 4, 6, \dots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{e}\}
\end{tikzcd}$$

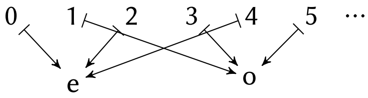
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $A_{\mathrm{eo}}(\mathrm{inc}) = \{(\mathrm{e}, \mathrm{o}), (\mathrm{o}, \mathrm{e})\} = A_{\mathrm{eo}}(\mathrm{dec})$

$$\begin{tikzcd}
\{0,2,4,\ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{1,3,5,\ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{o}\}
\end{tikzcd}
\qquad
\begin{tikzcd}
\{1,3,5,\ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{o}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{2,4,6,\ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{e}\}
\end{tikzcd}$$

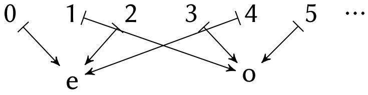
Approximation of data storage (example)

data storage Count:

- configurations: $\mathbb{N}$
- instructions: $\underbrace{\{(0,0)\}}_{(=0)}, \mathrm{inc}, \mathrm{dec} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

approximation strategy

$A_{\mathrm{eo}} \colon \mathbb{N} \dashrightarrow \{\mathrm{e}, \mathrm{o}\}$



data storage $A_{\mathrm{eo}}(\mathrm{Count})$:

- configurations: $\{\mathrm{e}, \mathrm{o}\}$
- instructions: $A_{\mathrm{eo}}((=0)) = \underbrace{\{(\mathrm{e}, \mathrm{e})\}}_{(=\mathrm{e})}$ $\quad A_{\mathrm{eo}}(\mathrm{inc}) = \underbrace{\{(\mathrm{e}, \mathrm{o}), (\mathrm{o}, \mathrm{e})\}}_{\text{toggle}} = A_{\mathrm{eo}}(\mathrm{dec})$

$$\begin{tikzcd}
\{0, 2, 4, \ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{e}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{1, 3, 5, \ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{o}\}
\end{tikzcd}
\qquad
\begin{tikzcd}
\{1, 3, 5, \ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] \arrow[d, mapsto, "\mathrm{inc}"'] & \{\mathrm{o}\} \arrow[d, mapsto, "A_{\mathrm{eo}}(\mathrm{inc})"] \\
\{2, 4, 6, \ldots\} \arrow[r, mapsto, "A_{\mathrm{eo}}"] & \{\mathrm{e}\}
\end{tikzcd}$$

Approximation of automata with data storage

Theorem (unweighted) [Denkinger 2017a, Thms. 21 and 26]

Let $\mathcal{M}$ be an (S, Σ)-automaton and A be an S-proper approximation strategy.

- If A is *total*, then $L(A(\mathcal{M})) \supseteq L(\mathcal{M})$.
- If A is *injective*, then $L(A(\mathcal{M})) \subseteq L(\mathcal{M})$.

Approximation of automata with data storage

Theorem (unweighted) [Denkinger 2017a, Thms. 21 and 26]

Let $\mathcal{M}$ be an (S, Σ)-automaton and A be an S-proper approximation strategy.

- If A is *total*, then $L(A(\mathcal{M})) \supseteq L(\mathcal{M})$.
- If A is *injective*, then $L(A(\mathcal{M})) \subseteq L(\mathcal{M})$.

Theorem (weighted) [Denkinger 2017a, Thm. 34]

Let $\mathcal{M}$ be an (S, Σ, K)-automaton, A be an S-proper approximation strategy, $\leq$ be a partial order on K, and K be positively $\leq$-ordered.

- If A is *total*, then $[\![A(\mathcal{M})]\!](w) \geq [\![\mathcal{M}]\!](w)$ for every $w \in \Sigma^*$.
- If A is *injective*, then $[\![A(\mathcal{M})]\!](w) \leq [\![\mathcal{M}]\!](w)$ for every $w \in \Sigma^*$.

Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w

Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w

$$(\mathcal{M}, w) \xmapsto{\text{2-best parse}}$$

Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w

$$\begin{tikzcd}
(\mathcal{M}, w) \arrow[r, mapsto, "\text{2-best parse}"] \arrow[d, mapsto, "A"'] & {} \\
(A(\mathcal{M}), w) &
\end{tikzcd}$$

Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w

$$\begin{tikzcd}
(\mathcal{M}, w) \arrow[rr, mapsto, "\text{2-best parse}"] \arrow[d, mapsto, "A"'] & & {} \\
(A(\mathcal{M}), w) \arrow[rr, mapsto, "m\text{-best parse}"] & & (r'_1, r'_2, r'_3, \ldots, r'_m)
\end{tikzcd}$$

decreasing weight

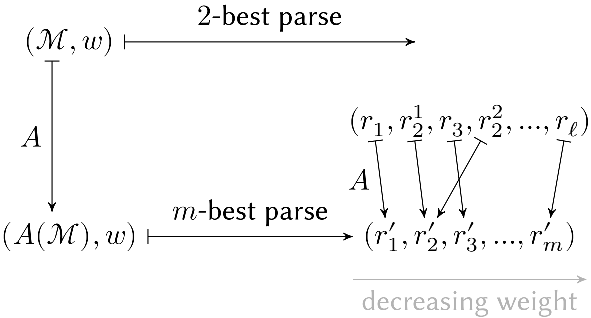
Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



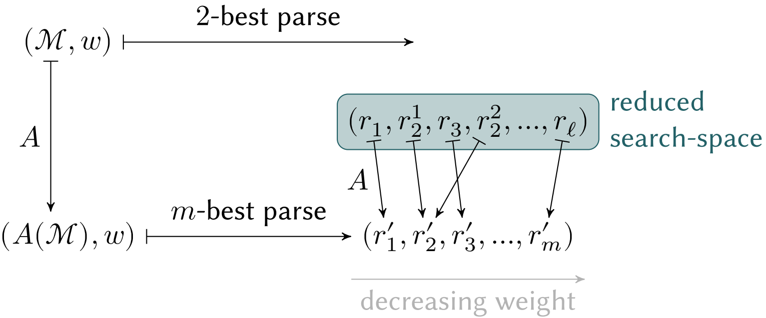
Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n, an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



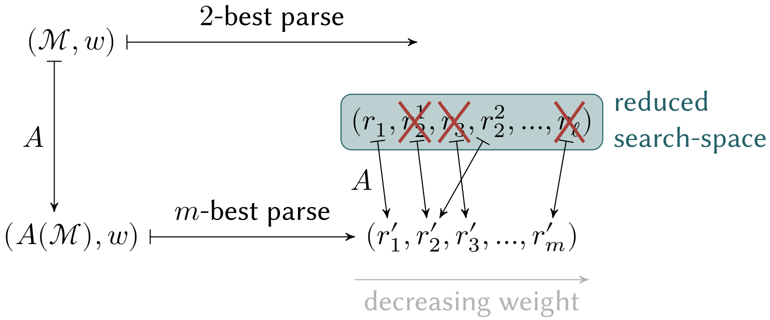
Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



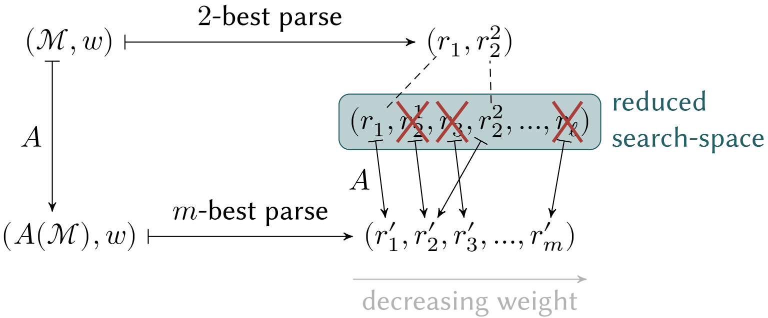
Coarse-to-fine parsing of weighted automata with data storage

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing [Denkinger 2017a, Alg. 3]

Input: an (S, Σ, K)-automaton $\mathcal{M}$, a word $w \in \Sigma$, a number n,
an S-proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



Publications in journals

T. Denkinger (2017b). “Chomsky-Schützenberger parsing for weighted multiple context-free languages”. *JLM.*

Publications in conference proceedings

T. Denkinger (2015). “A Chomsky-Schützenberger representation for weighted multiple context-free languages”. FSMNLP.

T. Denkinger (2016). “An Automata Characterisation for Multiple Context-Free Languages”. DLT.

T. Denkinger (2017a). “Approximation of Weighted Automata with Storage”. GandALF.

In preparation

T. Ruprecht and T. Denkinger. “Implementation of a Chomsky-Schützenberger n-best parser for weighted multiple context-free grammars”.

References I

N. Chomsky (1956). "Three models for the description of language". *IEEE Transactions on Information Theory.*

M. Droste and H. Vogler (2013). "The Chomsky-Schützenberger Theorem for Quantitative Context-Free Languages". Lecture Notes in Computer Science.

K. Evang and L. Kallmeyer (2011). "PLCFRS parsing of English discontinuous constituents". IWPT.

V. M. Jiménez and A. Marzal (2000). "Computation of the N Best Parse Trees for Weighted and Stochastic Context-Free Grammars".

M. Kuhlmann and J. Nivre (2006). "Mildly Non-projective Dependency Structures". COLING.

W. Maier and A. Søgaard (2008). "Treebanks and mild context-sensitivity".

D. Scott (1967). "Some definitional suggestions for automata theory". *JCSS.*

References II

H. Seki, T. Matsumura, M. Fujii, and T. Kasami (1991). “On multiple context-free grammars”. *TCS*.

É. Villemonte de la Clergerie (2002a). “Parsing MCS languages with thread automata”. TAG+.

É. Villemonte de la Clergerie (2002b). “Parsing Mildly Context-Sensitive Languages with Thread Automata”. COLING.

R. Yoshinaka, Y. Kaji, and H. Seki (2010). “Chomsky-Schützenberger-type characterization of multiple context-free languages”.