An automata characterisation for weighted multiple context-free languages

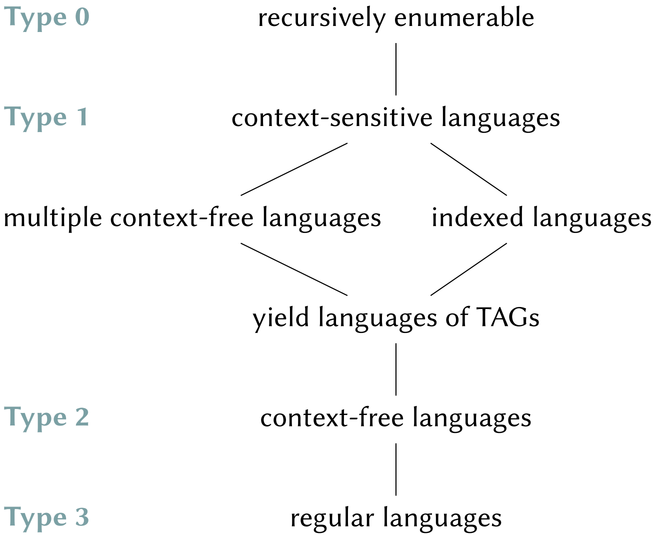
Tobias Denkinger

tobias.denkinger@tu-dresden.de

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

2016-04-25

A set diagram of some language classes



Type 0 recursively enumerable

Type 1 context-sensitive languages

multiple context-free languages

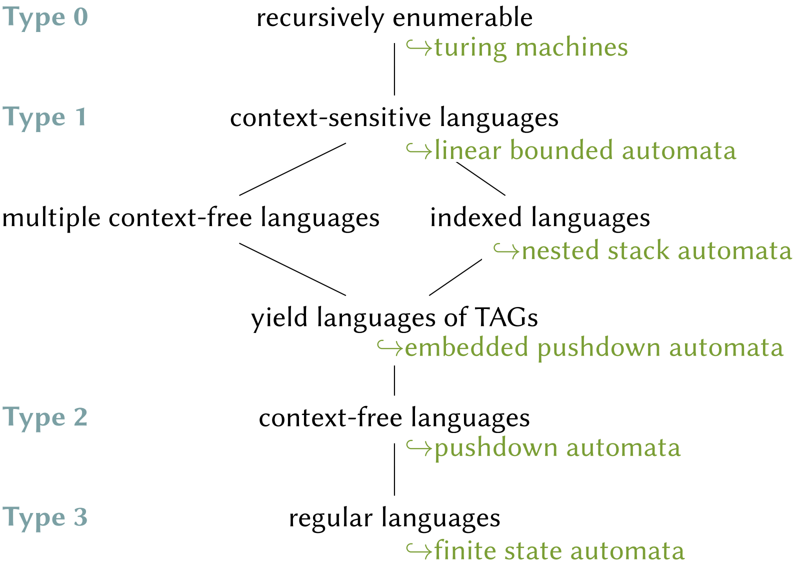
indexed languages

yield languages of TAGs

Type 2 context-free languages

Type 3 regular languages

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

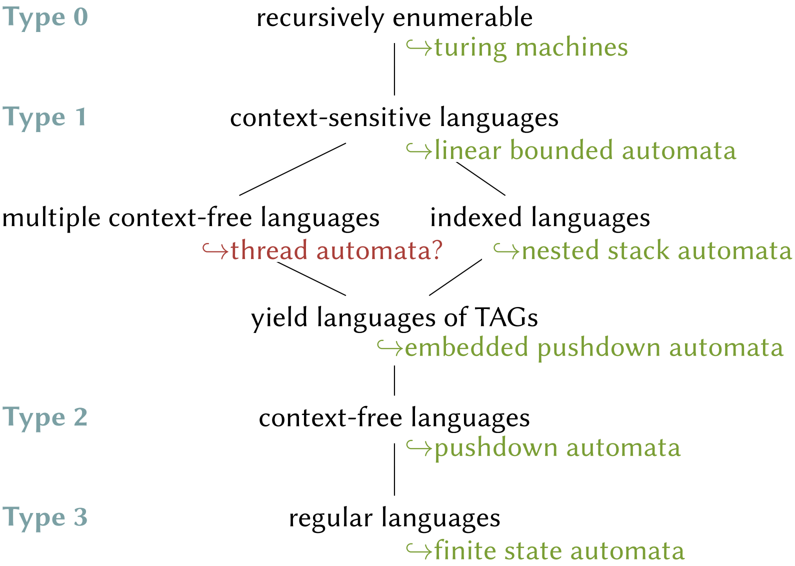
Type 2 context-free languages

↪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

↪thread 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

Outline

1. Weighted multiple context-free grammars

2. Weighted tree stack automata

3. The unweighted characterisation

4. The weighted characterisation

Composition functions

Example

$$(\Sigma^*) \times (\Sigma^* \times \Sigma^*) \to (\Sigma^* \times \Sigma^*)$$

Composition functions

Example

$$(\underset{\underset{x_1}{\downarrow}}{\Sigma^*}) \times (\underset{\underset{y_1}{\downarrow}}{\Sigma^*} \times \underset{\underset{y_2}{\downarrow}}{\Sigma^*}) \to (\Sigma^* \times \Sigma^*)$$

Composition functions

Example

$$[x_1 y_1, \beta y_2] \colon (\underset{\substack{\downarrow \\ x_1}}{\Sigma^*}) \times (\underset{\substack{\downarrow \\ y_1}}{\Sigma^*} \times \underset{\substack{\downarrow \\ y_2}}{\Sigma^*}) \to (\Sigma^* \times \Sigma^*)$$

Composition functions

Example

$$[x_1 y_1, \beta y_2] \colon (\underset{\underset{x_1}{\downarrow}}{\Sigma^*}) \times (\underset{\underset{y_1}{\downarrow}}{\Sigma^*} \times \underset{\underset{y_2}{\downarrow}}{\Sigma^*}) \to (\Sigma^* \times \Sigma^*)$$

$$[x_1 y_1, \beta y_2]((\alpha\gamma), (\alpha, \beta)) = (\alpha\gamma\alpha, \beta\beta)$$

Composition functions

Example

$$[x_1 y_1, \beta y_2] \colon (\underset{\underset{x_1}{\downarrow}}{\Sigma^*}) \times (\underset{\underset{y_1}{\downarrow}}{\Sigma^*} \times \underset{\underset{y_2}{\downarrow}}{\Sigma^*}) \to (\Sigma^* \times \Sigma^*)$$

$$[x_1 y_1, \beta y_2]((\alpha\gamma), (\alpha, \beta)) = (\alpha\gamma\alpha, \beta\beta)$$

Definition

$$[u_1, \dots, u_m] \colon (\Sigma^*)^{m_1} \times \cdots \times (\Sigma^*)^{m_k} \to (\Sigma^*)^m$$

$u_1, \dots, u_m \in (\Sigma \cup X)^*$, linear in the variables

Composition functions

Example

$$[x_1 y_1, \beta y_2] \colon (\underset{\underset{x_1}{\downarrow}}{\Sigma^*}) \times (\underset{\underset{y_1}{\downarrow}}{\Sigma^*} \times \underset{\underset{y_2}{\downarrow}}{\Sigma^*}) \to (\Sigma^* \times \Sigma^*)$$

$$[x_1 y_1, \beta y_2]((\alpha\gamma), (\alpha, \beta)) = (\alpha\gamma\alpha, \beta\beta)$$

Definition

$$[u_1, \dots, u_m] \colon (\Sigma^*)^{m_1} \times \cdots \times (\Sigma^*)^{m_k} \to (\Sigma^*)^m$$

$u_1, \dots, u_m \in (\Sigma \cup X)^*$, linear in the variables

$$[u_1, \dots, u_m]((w_1^1, \dots, w_1^{m_1}), \dots, (w_k^1, \dots, w_k^{m_k})) = (u'_1, \dots, u'_m)$$

u'_κ is obtained from u_κ by replacing x_i^j with w_i^j

Multiple context-free grammars (MCFGs)

productions:

Multiple context-free grammars (MCFGs)

productions:

$$\rho_1 = S \to [x_1 y_1 x_2 y_2](A, B)$$

$$\rho_2 = A \to [a x_1, c x_2](A)$$

$$\rho_3 = B \to [b x_1, d x_2](B)$$

$$\rho_4 = A \to [\varepsilon, \varepsilon]()$$

$$\rho_5 = B \to [\varepsilon, \varepsilon]()$$

Multiple context-free grammars (MCFGs)

productions:

$$\begin{aligned}
\rho_1 &= S \to [x_1 y_1 x_2 y_2](A, B)\\
\rho_2 &= A \to [a x_1, c x_2](A)\\
\rho_3 &= B \to [b x_1, d x_2](B)\\
\rho_4 &= A \to [\varepsilon, \varepsilon]()\\
\rho_5 &= B \to [\varepsilon, \varepsilon]()
\end{aligned}$$

2-multiple context-free grammar

Multiple context-free grammars (MCFGs)

productions:

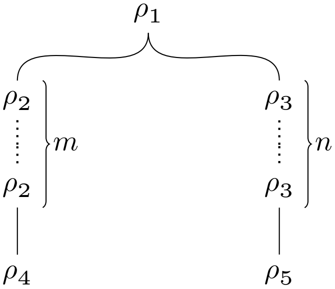
$$\rho_1 = S \to [x_1 y_1 x_2 y_2](A, B)$$

$$\rho_2 = A \to [a x_1, c x_2](A)$$

$$\rho_3 = B \to [b x_1, d x_2](B)$$

$$\rho_4 = A \to [\varepsilon, \varepsilon]()$$

$$\rho_5 = B \to [\varepsilon, \varepsilon]()$$

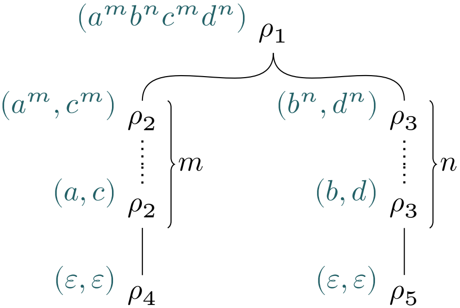


2-multiple context-free grammar

Multiple context-free grammars (MCFGs)

productions:

$$\rho_1 = S \to [x_1 y_1 x_2 y_2](A, B)$$
$$\rho_2 = A \to [a x_1, c x_2](A)$$
$$\rho_3 = B \to [b x_1, d x_2](B)$$
$$\rho_4 = A \to [\varepsilon, \varepsilon]()$$
$$\rho_5 = B \to [\varepsilon, \varepsilon]()$$



2-multiple context-free grammar

Multiple context-free grammars (MCFGs)

productions:

$$\rho_1 = S \to [x_1 y_1 x_2 y_2](A, B)$$
$$\rho_2 = A \to [a x_1, c x_2](A)$$
$$\rho_3 = B \to [b x_1, d x_2](B)$$
$$\rho_4 = A \to [\varepsilon, \varepsilon]()$$
$$\rho_5 = B \to [\varepsilon, \varepsilon]()$$

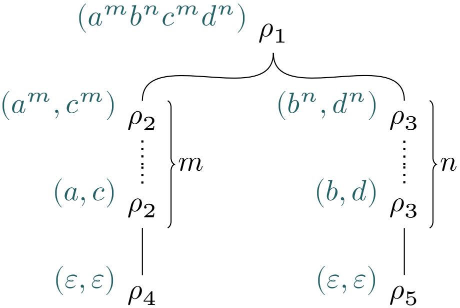
$\mu(\rho_1) = 1$

$\mu(\rho_2) = 1/2$

$\mu(\rho_3) = 1/3$

$\mu(\rho_4) = 1/2$

$\mu(\rho_5) = 2/3$



$([0,1], \max, \cdot, 0, 1)$-weighted 2-multiple context-free grammar

Multiple context-free grammars (MCFGs)

productions:

$$\rho_1 = S \to [x_1 y_1 x_2 y_2](A, B)$$

$$\rho_2 = A \to [a x_1, c x_2](A)$$

$$\rho_3 = B \to [b x_1, d x_2](B)$$

$$\rho_4 = A \to [\varepsilon, \varepsilon]()$$

$$\rho_5 = B \to [\varepsilon, \varepsilon]()$$

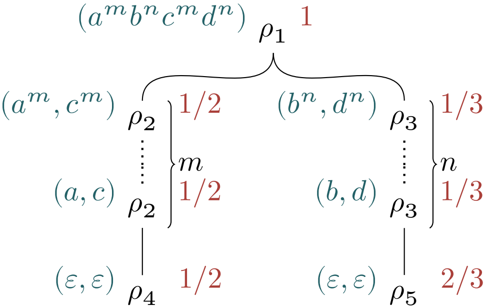
$$\mu(\rho_1) = 1$$

$$\mu(\rho_2) = 1/2$$

$$\mu(\rho_3) = 1/3$$

$$\mu(\rho_4) = 1/2$$

$$\mu(\rho_5) = 2/3$$



$([0,1], \max, \cdot, 0, 1)$-weighted 2-multiple context-free grammar

Multiple context-free grammars (MCFGs)

productions:

$$\rho_1 = S \to [x_1 y_1 x_2 y_2](A, B)$$
$$\rho_2 = A \to [a x_1, c x_2](A)$$
$$\rho_3 = B \to [b x_1, d x_2](B)$$
$$\rho_4 = A \to [\varepsilon, \varepsilon]()$$
$$\rho_5 = B \to [\varepsilon, \varepsilon]()$$

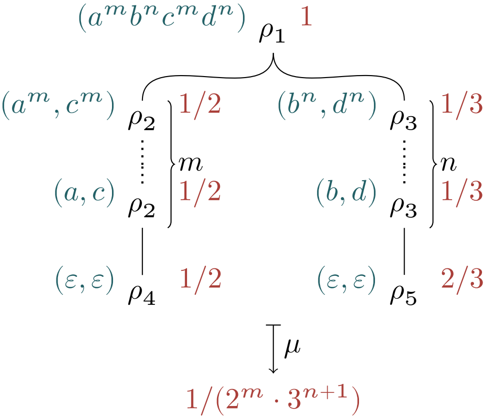
$\mu(\rho_1) = 1$

$\mu(\rho_2) = 1/2$

$\mu(\rho_3) = 1/3$

$\mu(\rho_4) = 1/2$

$\mu(\rho_5) = 2/3$



$(a^m b^n c^m d^n)$ ρ_1 1
(a^m, c^m) ρ_2 $1/2$
(b^n, d^n) ρ_3 $1/3$
m
n
(a, c) ρ_2 $1/2$
(b, d) ρ_3 $1/3$
$(\varepsilon, \varepsilon)$ ρ_4 $1/2$
$(\varepsilon, \varepsilon)$ ρ_5 $2/3$
μ
$1/(2^m \cdot 3^{n+1})$

$([0,1], \max, \cdot, 0, 1)$-weighted 2-multiple context-free grammar

Weight algebras

complete commutative strong bimonoids

- “strong bimonoid = semiring without requiring distributivity”
- multiplication is (also) commutative
- $\sum$-operation also for infinite index sets

Weight algebras

complete commutative strong bimonoids

- “strong bimonoid = semiring without requiring distributivity”
- multiplication is (also) commutative
- $\sum$-operation also for infinite index sets

examples:

Weight algebras

complete commutative strong bimonoids

- "strong bimonoid = semiring without requiring distributivity"
- multiplication is (also) commutative
- $\sum$-operation also for infinite index sets

examples:

- all complete commutative semirings (Boolean semiring, probability semiring, Viterbi semiring, ...)

Weight algebras

complete commutative strong bimonoids

- "strong bimonoid = semiring without requiring distributivity"
- multiplication is (also) commutative
- $\sum$-operation also for infinite index sets

examples:

- all complete commutative semirings (Boolean semiring, probability semiring, Viterbi semiring, ...)
- all complete lattices

Weight algebras

complete commutative strong bimonoids

- “strong bimonoid = semiring without requiring distributivity”
- multiplication is (also) commutative
- $\sum$-operation also for infinite index sets

examples:

- all complete commutative semirings (Boolean semiring, probability semiring, Viterbi semiring, ...)
- all complete lattices
- tropical bimonoid: $(\mathbb{R}_{\geq 0}^{\infty}, +, \min, 0, \infty)$

The tree stack (idea from Villemonte de la Clergerie 2002)

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

- stack symbols Γ

The tree stack (idea from Villemonte de la Clergerie 2002)

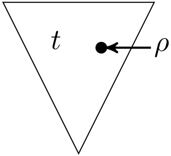
data type $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function
  $t\colon \mathbb{N}_+^* \to \Gamma$
- stack pointer $\rho \in \mathbb{N}_+^*$
  from the domain of t

The tree stack (idea from Villemonte de la Clergerie 2002)

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

- stack symbols Γ
- partial function $t\colon \mathbb{N}_+^* \to \Gamma$
- domain of t prefix-closed (but not necessarily sibling-closed)

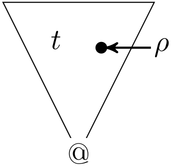


- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

The tree stack (idea from Villemonte de la Clergerie 2002)

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

- stack symbols Γ
- partial function $t\colon \mathbb{N}_+^* \to \Gamma \uplus \{@\}$
- domain of t prefix-closed (but not necessarily sibling-closed)

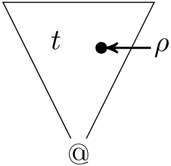


- $t(\varepsilon) = @$
- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

The tree stack (idea from Villemonte de la Clergerie 2002)

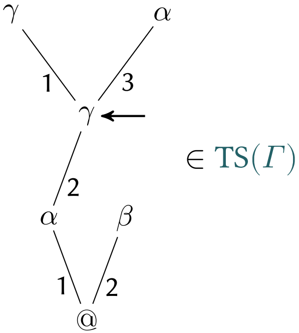
data type $\mathrm{TS}(\Gamma)$

- stack symbols Γ
- partial function $t\colon \mathbb{N}_+^* \to \Gamma \uplus \{@\}$
- domain of t prefix-closed (but not necessarily sibling-closed)



- $t(\varepsilon) = @$
- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

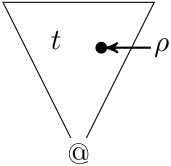
example



The tree stack (idea from Villemonte de la Clergerie 2002)

data type $\mathrm{TS}(\mathit{\Gamma})$

- stack symbols $\mathit{\Gamma}$
- partial function $t\colon \mathbb{N}_+^* \to \mathit{\Gamma} \uplus \{@\}$
- domain of t prefix-closed (but not necessarily sibling-closed)



- $t(\varepsilon) = @$
- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

***predicates* (unary)**

$$\mathrm{true} = \mathrm{TS}(\mathit{\Gamma})$$

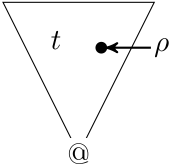
$$\mathrm{bot} = \{(t, \rho) \in \mathrm{TS}(\mathit{\Gamma}) \mid \rho = \varepsilon\}$$

$$\mathrm{eq}(\gamma) = \{(t, \rho) \in \mathrm{TS}(\mathit{\Gamma}) \mid t(\rho) = \gamma\}$$

The tree stack (idea from Villemonte de la Clergerie 2002)

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

- stack symbols Γ
- partial function $t\colon \mathbb{N}_+^* \to \Gamma \uplus \{@\}$
- domain of t prefix-closed (but not necessarily sibling-closed)



- $t(\varepsilon) = @$
- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

***predicates* (unary)**

$\mathrm{true} = \mathrm{TS}(\Gamma)$

$\mathrm{bot} = \{(t, \rho) \in \mathrm{TS}(\Gamma) \mid \rho = \varepsilon\}$

$\mathrm{eq}(\gamma) = \{(t, \rho) \in \mathrm{TS}(\Gamma) \mid t(\rho) = \gamma\}$

instructions

id

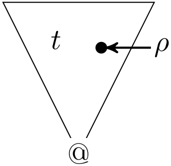
$\mathrm{set}(\gamma)$: $t(\rho) := \gamma$
(only if $\rho \neq \varepsilon$)

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

The tree stack (idea from Villemonte de la Clergerie 2002)

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

- stack symbols Γ
- partial function $t\colon \mathbb{N}_+^* \to \Gamma \uplus \{@\}$
- domain of t prefix-closed (but not necessarily sibling-closed)



- $t(\varepsilon) = @$
- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

predicates **(unary)**

$\mathrm{true} = \mathrm{TS}(\Gamma)$

$\mathrm{bot} = \{(t, \rho) \in \mathrm{TS}(\Gamma) \mid \rho = \varepsilon\}$

$\mathrm{eq}(\gamma) = \{(t, \rho) \in \mathrm{TS}(\Gamma) \mid t(\rho) = \gamma\}$

instructions

id

$\mathrm{set}(\gamma)$: $t(\rho) := \gamma$
(only if $\rho \neq \varepsilon$)

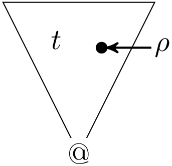
up_i: move stack pointer to i-th child
(only if $t(\rho i)$ is defined)

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

The tree stack (idea from Villemonte de la Clergerie 2002)

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

- stack symbols Γ
- partial function $t\colon \mathbb{N}_+^* \to \Gamma \uplus \{@\}$
- domain of t prefix-closed (but not necessarily sibling-closed)



- $t(\varepsilon) = @$
- stack pointer $\rho \in \mathbb{N}_+^*$ from the domain of t

***predicates* (unary)**

$$\mathrm{true} = \mathrm{TS}(\Gamma)$$

$$\mathrm{bot} = \{(t, \rho) \in \mathrm{TS}(\Gamma) \mid \rho = \varepsilon\}$$

$$\mathrm{eq}(\gamma) = \{(t, \rho) \in \mathrm{TS}(\Gamma) \mid t(\rho) = \gamma\}$$

instructions

id

$\mathrm{set}(\gamma)$: $t(\rho) := \gamma$
(only if $\rho \neq \varepsilon$)

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

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

down: move stack pointer to parent

Tree-stack automata (TSA) as automata with storage

symbols read

transitions:

$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \qquad\qquad\qquad\qquad , 1)\\
\tau_2 &= (1, \varepsilon, \qquad\qquad\qquad\qquad , 2)\\
\tau_3 &= (2, \varepsilon, \qquad\qquad\qquad\qquad , 2)\\
\tau_4 &= (2, \mathrm{b}, \qquad\qquad\qquad\qquad , 2)\\
\tau_5 &= (2, \varepsilon, \qquad\qquad\qquad\qquad , 3)\\
\tau_6 &= (3, \mathrm{c}, \qquad\qquad\qquad\qquad , 3)\\
\tau_7 &= (3, \varepsilon, \qquad\qquad\qquad\qquad , 4)\\
\tau_8 &= (4, \mathrm{d}, \qquad\qquad\qquad\qquad , 4)\\
\tau_9 &= (4, \varepsilon, \qquad\qquad\qquad\qquad , 5)
\end{aligned}
$$

source and target states

Tree-stack automata (TSA) as automata with storage

symbols read

transitions:

instructions

$$
\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true}, & \mathrm{push}_1(*), & 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true}, & \mathrm{push}_1(\#), & 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#), & \mathrm{down}, & 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*), & \mathrm{down}, & 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot}, & \mathrm{up}_1, & 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*), & \mathrm{up}_1, & 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#), & \mathrm{down}, & 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*), & \mathrm{down}, & 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot}, & \mathrm{id}, & 5)
\end{array}
$$

predicates

source and target states

Tree-stack automata (TSA) as automata with storage

transitions:

$$
\begin{array}{lllllll}
\tau_1 = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}
$$

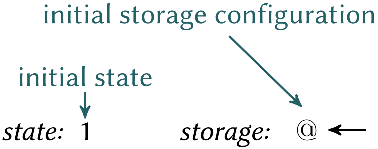
language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

Tree-stack automata (TSA) as automata with storage

transitions:

$$\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$



initial storage configuration

initial state

state: 1

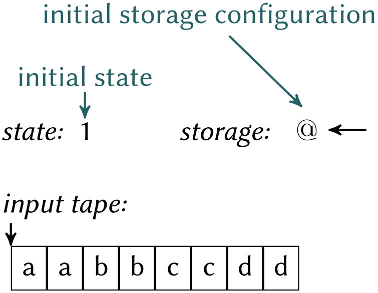
storage: @ ←

Tree-stack automata (TSA) as automata with storage

transitions:

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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$



initial storage configuration

initial state

state: 1

storage: @

input tape:

a	a	b	b	c	c	d	d

Tree-stack automata (TSA) as automata with storage

transitions:

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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 1 *storage:* @ ←

input tape:

↓

a	a	b	b	c	c	d	d

run:

Tree-stack automata (TSA) as automata with storage

transitions:

$$\tau_1 = (1\,,\mathrm{a}\,,\mathrm{true}\,,\mathrm{push}_1(*)\,,1)$$

$$\tau_2 = (1\,,\varepsilon\,,\mathrm{true}\,,\mathrm{push}_1(\#)\,,2)$$

$$\tau_3 = (2\,,\varepsilon\,,\mathrm{eq}(\#)\,,\mathrm{down}\,,2)$$

$$\tau_4 = (2\,,\mathrm{b}\,,\mathrm{eq}(*)\,,\mathrm{down}\,,2)$$

$$\tau_5 = (2\,,\varepsilon\,,\mathrm{bot}\,,\mathrm{up}_1\,,3)$$

$$\tau_6 = (3\,,\mathrm{c}\,,\mathrm{eq}(*)\,,\mathrm{up}_1\,,3)$$

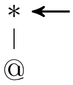
$$\tau_7 = (3\,,\varepsilon\,,\mathrm{eq}(\#)\,,\mathrm{down}\,,4)$$

$$\tau_8 = (4\,,\mathrm{d}\,,\mathrm{eq}(*)\,,\mathrm{down}\,,4)$$

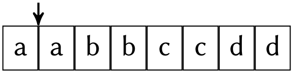
$$\tau_9 = (4\,,\varepsilon\,,\mathrm{bot}\,,\mathrm{id}\,,5)$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 1 *storage:*



input tape:



a	a	b	b	c	c	d	d

run:

τ_1

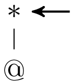
Tree-stack automata (TSA) as automata with storage

transitions:

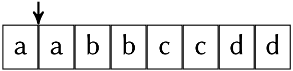
$$
\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}
$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 1 *storage:*



input tape:



run:

τ_1

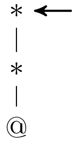
Tree-stack automata (TSA) as automata with storage

transitions:

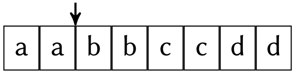
$$\begin{aligned}
\tau_1 &= (1\,, \mathrm{a}\,, \mathrm{true} &&, \mathrm{push}_1(*) &&, 1) \\
\tau_2 &= (1\,, \varepsilon\,, \mathrm{true} &&, \mathrm{push}_1(\#) &&, 2) \\
\tau_3 &= (2\,, \varepsilon\,, \mathrm{eq}(\#) &&, \mathrm{down} &&, 2) \\
\tau_4 &= (2\,, \mathrm{b}\,, \mathrm{eq}(*) &&, \mathrm{down}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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 1 *storage:*



input tape:



a	a	b	b	c	c	d	d

run:

$\tau_1 \tau_1$

Tree-stack automata (TSA) as automata with storage

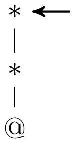
transitions:

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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 1

storage:



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1 \tau_1$

Tree-stack automata (TSA) as automata with storage

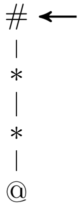
transitions:

$$\tau_1 = (1\,,\mathrm{a}\,,\mathrm{true}\,,\mathrm{push}_1(*)\,,1)$$
$$\tau_2 = (1\,,\varepsilon\,,\mathrm{true}\,,\mathrm{push}_1(\#)\,,2)$$
$$\tau_3 = (2\,,\varepsilon\,,\mathrm{eq}(\#)\,,\mathrm{down}\,,2)$$
$$\tau_4 = (2\,,\mathrm{b}\,,\mathrm{eq}(*)\,,\mathrm{down}\,,2)$$
$$\tau_5 = (2\,,\varepsilon\,,\mathrm{bot}\,,\mathrm{up}_1\,,3)$$
$$\tau_6 = (3\,,\mathrm{c}\,,\mathrm{eq}(*)\,,\mathrm{up}_1\,,3)$$
$$\tau_7 = (3\,,\varepsilon\,,\mathrm{eq}(\#)\,,\mathrm{down}\,,4)$$
$$\tau_8 = (4\,,\mathrm{d}\,,\mathrm{eq}(*)\,,\mathrm{down}\,,4)$$
$$\tau_9 = (4\,,\varepsilon\,,\mathrm{bot}\,,\mathrm{id}\,,5)$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2

storage:



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2$

Tree-stack automata (TSA) as automata with storage

transitions:

$\tau_1 = (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)$

$\tau_2 = (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)$

$\tau_3 = (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)$

$\tau_4 = (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)$

$\tau_5 = (2, \varepsilon, \mathrm{bot}, \mathrm{up}_1, 3)$

$\tau_6 = (3, \mathrm{c}, \mathrm{eq}(*), \mathrm{up}_1, 3)$

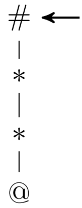
$\tau_7 = (3, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 4)$

$\tau_8 = (4, \mathrm{d}, \mathrm{eq}(*), \mathrm{down}, 4)$

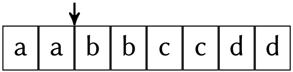
$\tau_9 = (4, \varepsilon, \mathrm{bot}, \mathrm{id}, 5)$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2 *storage:*



input tape:



a	a	b	b	c	c	d	d

run:

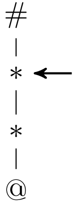
$\tau_1 \tau_1 \tau_2$

Tree-stack automata (TSA) as automata with storage

transitions:

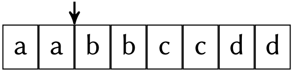
$$\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)\\
\tau_5 &= (2, \varepsilon,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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$



state: 2 *storage:*

input tape:



a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3$

Tree-stack automata (TSA) as automata with storage

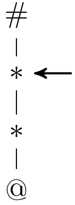
transitions:

$$\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2

storage:



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1 \tau_1 \tau_2 \tau_3$

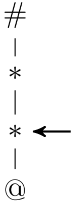
Tree-stack automata (TSA) as automata with storage

transitions:

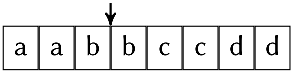
$$
\begin{aligned}
\tau_1 &= (1\,, \mathrm{a}\,, \mathrm{true} &&, \mathrm{push}_1(*) &&, 1)\\
\tau_2 &= (1\,, \varepsilon\,, \mathrm{true} &&, \mathrm{push}_1(\#) &&, 2)\\
\tau_3 &= (2\,, \varepsilon\,, \mathrm{eq}(\#) &&, \mathrm{down} &&, 2)\\
\tau_4 &= (2\,, \mathrm{b}\,, \mathrm{eq}(*) &&, \mathrm{down}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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2 *storage:*



input tape:



a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4$

Tree-stack automata (TSA) as automata with storage

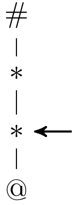
transitions:

$$\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}$$

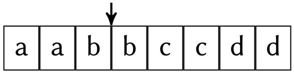
language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2

storage:



input tape:



a	a	b	b	c	c	d	d

run:

$\tau_1 \tau_1 \tau_2 \tau_3 \tau_4$

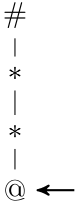
Tree-stack automata (TSA) as automata with storage

transitions:

$$
\begin{array}{llllll}
\tau_1 = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}
$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2 *storage:*



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4$

Tree-stack automata (TSA) as automata with storage

transitions:

$\tau_1 = (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)$

$\tau_2 = (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)$

$\tau_3 = (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)$

$\tau_4 = (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)$

$\tau_5 = (2, \varepsilon, \mathrm{bot}, \mathrm{up}_1, 3)$

$\tau_6 = (3, \mathrm{c}, \mathrm{eq}(*), \mathrm{up}_1, 3)$

$\tau_7 = (3, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 4)$

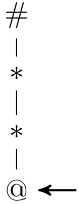
$\tau_8 = (4, \mathrm{d}, \mathrm{eq}(*), \mathrm{down}, 4)$

$\tau_9 = (4, \varepsilon, \mathrm{bot}, \mathrm{id}, 5)$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 2

storage:



input tape:

				↓			
a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4$

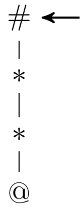
Tree-stack automata (TSA) as automata with storage

transitions:

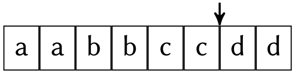
$$\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 3 *storage:*



input tape:



a	a	b	b	c	c	d	d

run:

$\tau_1 \tau_1 \tau_2 \tau_3 \tau_4 \tau_4 \tau_5 \tau_5 \tau_6$

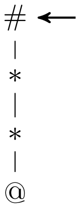
Tree-stack automata (TSA) as automata with storage

transitions:

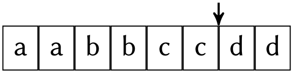
$$\begin{array}{lllllll}
\tau_1 & = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 & = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 & = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 & = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 & = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 & = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 & = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 & = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 & = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 3 *storage:*



input tape:



a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4\tau_5\tau_5\tau_6$

Tree-stack automata (TSA) as automata with storage

transitions:

$$\tau_1 = (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)$$

$$\tau_2 = (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)$$

$$\tau_3 = (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)$$

$$\tau_4 = (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)$$

$$\tau_5 = (2, \varepsilon, \mathrm{bot}, \mathrm{up}_1, 3)$$

$$\tau_6 = (3, \mathrm{c}, \mathrm{eq}(*), \mathrm{up}_1, 3)$$

$$\tau_7 = (3, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 4)$$

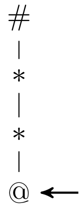
$$\tau_8 = (4, \mathrm{d}, \mathrm{eq}(*), \mathrm{down}, 4)$$

$$\tau_9 = (4, \varepsilon, \mathrm{bot}, \mathrm{id}, 5)$$

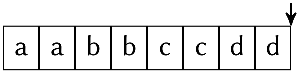
language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 4

storage:



input tape:



run:

$\tau_1 \tau_1 \tau_2 \tau_3 \tau_4 \tau_4 \tau_5 \tau_5 \tau_6 \tau_7 \tau_8 \tau_8$

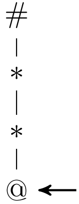
Tree-stack automata (TSA) as automata with storage

transitions:

$$\begin{array}{lll}
\tau_1 & = & (1\,, \mathrm{a}\,, \mathrm{true} \quad, \mathrm{push}_1(*) \;\;, 1) \\
\tau_2 & = & (1\,, \varepsilon\,, \mathrm{true} \quad, \mathrm{push}_1(\#)\,, 2) \\
\tau_3 & = & (2\,, \varepsilon\,, \mathrm{eq}(\#)\,, \mathrm{down} \qquad, 2) \\
\tau_4 & = & (2\,, \mathrm{b}\,, \mathrm{eq}(*) \;\,, \mathrm{down} \qquad, 2) \\
\tau_5 & = & (2\,, \varepsilon\,, \mathrm{bot} \quad\;\, , \mathrm{up}_1 \qquad\quad, 3) \\
\tau_6 & = & (3\,, \mathrm{c}\,, \mathrm{eq}(*) \;\,, \mathrm{up}_1 \qquad\quad, 3) \\
\tau_7 & = & (3\,, \varepsilon\,, \mathrm{eq}(\#)\,, \mathrm{down} \qquad, 4) \\
\tau_8 & = & (4\,, \mathrm{d}\,, \mathrm{eq}(*) \;\,, \mathrm{down} \qquad, 4) \\
\tau_9 & = & (4\,, \varepsilon\,, \mathrm{bot} \quad\;\, , \mathrm{id} \qquad\quad\;\, , 5)
\end{array}$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 4 *storage:*



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4\tau_5\tau_5\tau_6\tau_7\tau_8\tau_8$

Tree-stack automata (TSA) as automata with storage

transitions:

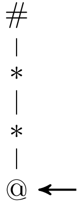
$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)\\
\tau_5 &= (2, \varepsilon,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

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

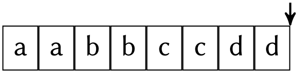
final state

state: 5

storage:



input tape:



run:

$\tau_1 \tau_1 \tau_2 \tau_3 \tau_4 \tau_4 \tau_5 \tau_5 \tau_6 \tau_7 \tau_8 \tau_8 \tau_9$

Tree-stack automata (TSA) as automata with storage

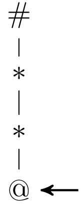
transitions:

$$
\begin{array}{llllll}
\tau_1 = (1, & \mathrm{a}, & \mathrm{true} & , \mathrm{push}_1(*) & , 1) \\
\tau_2 = (1, & \varepsilon, & \mathrm{true} & , \mathrm{push}_1(\#) & , 2) \\
\tau_3 = (2, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 2) \\
\tau_4 = (2, & \mathrm{b}, & \mathrm{eq}(*) & , \mathrm{down} & , 2) \\
\tau_5 = (2, & \varepsilon, & \mathrm{bot} & , \mathrm{up}_1 & , 3) \\
\tau_6 = (3, & \mathrm{c}, & \mathrm{eq}(*) & , \mathrm{up}_1 & , 3) \\
\tau_7 = (3, & \varepsilon, & \mathrm{eq}(\#) & , \mathrm{down} & , 4) \\
\tau_8 = (4, & \mathrm{d}, & \mathrm{eq}(*) & , \mathrm{down} & , 4) \\
\tau_9 = (4, & \varepsilon, & \mathrm{bot} & , \mathrm{id} & , 5)
\end{array}
$$

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 5

storage:



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4\tau_5\tau_5\tau_6\tau_7\tau_8\tau_8\tau_9$

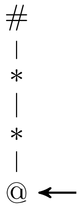
2-restricted: enter each stack position at most 2 times *from below*

Tree-stack automata (TSA) as automata with storage

transitions:	μ
$\tau_1 = (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)$	$1/2$
$\tau_2 = (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)$	$1/2$
$\tau_3 = (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)$	1
$\tau_4 = (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)$	1
$\tau_5 = (2, \varepsilon, \mathrm{bot}, \mathrm{up}_1, 3)$	1
$\tau_6 = (3, \mathrm{c}, \mathrm{eq}(*), \mathrm{up}_1, 3)$	$1/3$
$\tau_7 = (3, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 4)$	$2/3$
$\tau_8 = (4, \mathrm{d}, \mathrm{eq}(*), \mathrm{down}, 4)$	1
$\tau_9 = (4, \varepsilon, \mathrm{bot}, \mathrm{id}, 5)$	1

language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 5 *storage:*



input tape:

a	a	b	b	c	c	d	d

run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4\tau_5\tau_5\tau_6\tau_7\tau_8\tau_8\tau_9$

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

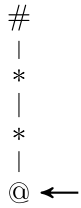
Tree-stack automata (TSA) as automata with storage

transitions:	μ
$\tau_1 = (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)$	$1/2$
$\tau_2 = (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)$	$1/2$
$\tau_3 = (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)$	1
$\tau_4 = (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)$	1
$\tau_5 = (2, \varepsilon, \mathrm{bot}, \mathrm{up}_1, 3)$	1
$\tau_6 = (3, \mathrm{c}, \mathrm{eq}(*), \mathrm{up}_1, 3)$	$1/3$
$\tau_7 = (3, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 4)$	$2/3$
$\tau_8 = (4, \mathrm{d}, \mathrm{eq}(*), \mathrm{down}, 4)$	1
$\tau_9 = (4, \varepsilon, \mathrm{bot}, \mathrm{id}, 5)$	1

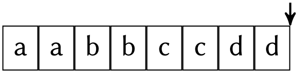
language: $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$

state: 5

storage:



input tape:



run:

$\tau_1\tau_1\tau_2\tau_3\tau_4\tau_4\tau_5\tau_5\tau_6\tau_7\tau_8\tau_8\tau_9$

weight: $1/(2^2 \cdot 3^3)$

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

The unweighted characterisation

Theorem (new)

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

Proof sketch. Show both set inclusions by construction.

k-MCFL: languages generated by k-MCFGs

k-TSL: languages recognised by k-restricted tree stack automata

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

Lemma (new)

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

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

Lemma (new)

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

Construction sketch.

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

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

Lemma (new)

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

Construction sketch.

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

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

Lemma (new)

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

Construction sketch.

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

example transitions:

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

Lemma (new)

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

Construction sketch.

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

example transitions:

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

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

Lemma (new)

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

Construction sketch.

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

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

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

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

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

Lemma (new)

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

Construction sketch.

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

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

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

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

return: $(\langle\rho,1,2\rangle, \varepsilon, \text{eq}(\langle\bar{\rho},i,j\rangle), \text{set}(\rho), \langle\bar{\rho},i,j\rangle_{-})$
$(\langle\bar{\rho},i,j\rangle_{-}, \varepsilon, \text{true}, \text{down}, \langle\bar{\rho},i,j\rangle)$

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

Lemma (new)

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

Construction sketch.

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

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

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

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

return: $(\langle\rho,1,2\rangle, \varepsilon, \text{eq}(\langle\bar{\rho},i,j\rangle), \text{set}(\rho), \langle\bar{\rho},i,j\rangle_-)$
$(\langle\bar{\rho},i,j\rangle_-, \varepsilon, \text{true}, \text{down}, \langle\bar{\rho},i,j\rangle)$

resume: $(\langle\rho,2,1\rangle, \varepsilon, \text{true}, \text{up}_1, \langle\rho,2,1\rangle_+)$
$(\langle\rho,2,1\rangle_+, \varepsilon, \text{eq}(\rho'), \text{set}(\langle\rho,2,2\rangle), \langle\rho',2,0\rangle)$

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

Lemma (new)

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

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

Lemma (new)

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

Proof idea.

(1) construct an MCFG that generates the runs

(2) use closure of MCFG under homomorphisms

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

Lemma (new)

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

Proof idea.

(1) construct an MCFG that generates the runs

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

if and only if

- $\theta_1, \dots, \theta_m$ 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 MCFG under homomorphisms

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

transitions:

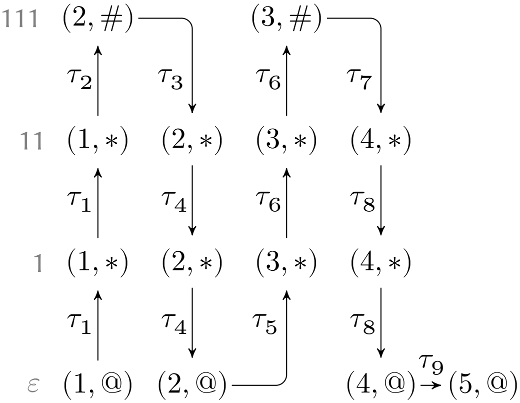
$$
\begin{aligned}
\tau_1 &= (1, \mathrm{a}, \mathrm{true}, \mathrm{push}_1(*), 1)\\
\tau_2 &= (1, \varepsilon, \mathrm{true}, \mathrm{push}_1(\#), 2)\\
\tau_3 &= (2, \varepsilon, \mathrm{eq}(\#), \mathrm{down}, 2)\\
\tau_4 &= (2, \mathrm{b}, \mathrm{eq}(*), \mathrm{down}, 2)\\
\tau_5 &= (2, \varepsilon,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

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

transitions:

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

run and the stack:

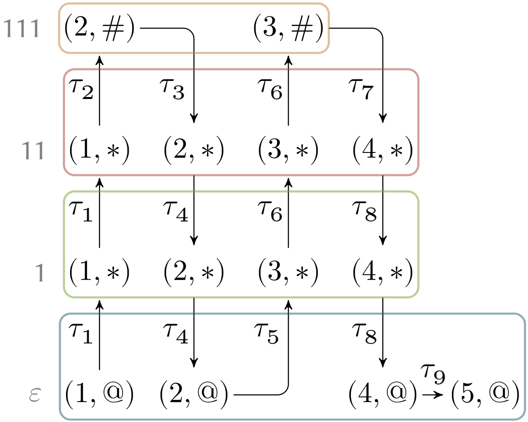


k-TSL$_\mathrm{r}$ $\subseteq$ k-MCFL (example)

transitions:

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

run and the stack:



111 $(2,\#)$ $(3,\#)$

τ_2 τ_3 τ_6 τ_7

11 $(1,*)$ $(2,*)$ $(3,*)$ $(4,*)$

τ_1 τ_4 τ_6 τ_8

1 $(1,*)$ $(2,*)$ $(3,*)$ $(4,*)$

τ_1 τ_4 τ_5 τ_8

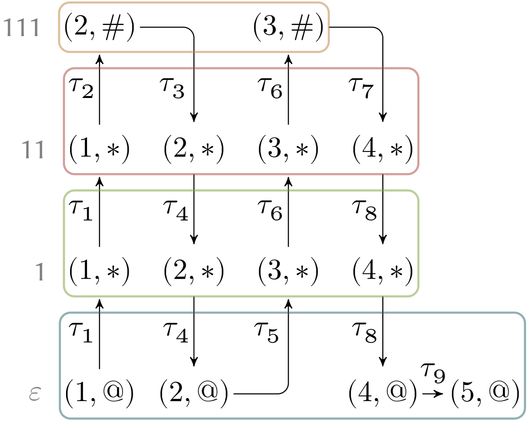
ε $(1,@)$ $(2,@)$ $(4,@) \xrightarrow{\tau_9} (5,@)$

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

transitions:

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

run and the stack:



rules:

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

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

The weighted characterisation

Corollary (new)

For every complete commutative strong bimonoid $\mathcal{A}$ we have

$$k\text{-MCFL}_{\mathcal{A}} = k\text{-TSL}_{\mathsf{r},\mathcal{A}}$$

k-MCFL$_{\mathcal{A}}$: languages generated by $\mathcal{A}$-weighted k-MCFGs

k-TSL$_{\mathsf{r},\mathcal{A}}$: languages recognised by $\mathcal{A}$-weighted k-restricted TSA

The weighted characterisation

Corollary (new)

For every complete commutative strong bimonoid $\mathcal{A}$ we have

$$k\text{-MCFL}_{\mathcal{A}} = k\text{-TSL}_{\mathsf{r},\mathcal{A}}$$

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{\mathsf{r},\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_{\mathsf{r}})$$

$k\text{-MCFL}_{\mathcal{A}}$: languages generated by $\mathcal{A}$-weighted k-MCFGs

$k\text{-TSL}_{\mathsf{r}}$ / $k\text{-TSL}_{\mathsf{r},\mathcal{A}}$: languages recognised by ($\mathcal{A}$-weighted) k-restricted TSA

$\alpha\text{HOM}_{\mathcal{A}}$: alphabetic $\mathcal{A}$-weighted homomorphisms

The weighted characterisation

Corollary (new)

For every complete commutative strong bimonoid $\mathcal{A}$ we have

$$k\text{-MCFL}_{\mathcal{A}} = k\text{-TSL}_{\mathsf{r},\mathcal{A}}$$

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{\mathsf{r},\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_{\mathsf{r}})$$

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

---

k-MCFL / k-MCFL$_{\mathcal{A}}$: languages generated by ($\mathcal{A}$-weighted) k-MCFGs

k-TSL$_{\mathsf{r}}$ / k-TSL$_{\mathsf{r},\mathcal{A}}$: languages recognised by ($\mathcal{A}$-weighted) k-restricted TSA

αHOM$_{\mathcal{A}}$: alphabetic $\mathcal{A}$-weighted homomorphisms

The weighted characterisation

Corollary (new)

For every complete commutative strong bimonoid $\mathcal{A}$ we have

$$k\text{-MCFL}_{\mathcal{A}} = k\text{-TSL}_{\mathsf{r},\mathcal{A}}$$

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{\mathsf{r},\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_{\mathsf{r}})$$

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

Theorem (new)

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

References

[1] É. Villemonte de la Clergerie. “Parsing Mildly Context-Sensitive Languages with Thread Automata”. 2002.

[2] L. Herrmann and H. Vogler. “A Chomsky-Schützenberger Theorem for Weighted Automata with Storage”. 2015.

[3] T. Denkinger. “A Chomsky-Schützenberger representation for weighted multiple context-free languages”. 2015.

Weight separation for weighted automata with storage

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{r,\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_r)$$

Construction sketch.

$\mathcal{M}\colon \tau = (q, \omega, p, f, q')$

Weight separation for weighted automata with storage

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{r,\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_r)$$

Construction sketch.

$$\begin{tikzcd}
 & \mathcal{M}\colon \tau = (q, \omega, p, f, q') \arrow[dl, "\mu"'] \arrow[dr, "\text{reads}"] & \\
\mu(\tau) & & \omega
\end{tikzcd}$$

Weight separation for weighted automata with storage

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{r,\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_r)$$

Construction sketch.

$$\begin{tikzcd}
\mathcal{M}\colon \tau = (q, \omega, p, f, q') \arrow[rr, "\text{construction}"] \arrow[d, "\mu"'] \arrow[dr, "\text{reads}"] & & \mathcal{M}'\colon (q, \tau, p, f, q') \\
\mu(\tau) & \omega &
\end{tikzcd}$$

Weight separation for weighted automata with storage

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{r,\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_r)$$

Construction sketch.

$$\begin{tikzcd}
{} & \mathcal{M}\colon \tau = (q, \omega, p, f, q') \arrow[dl, "\mu"'] \arrow[dr, "\text{reads}"] \arrow[rr, "\text{construction}"] & {} & \mathcal{M}'\colon (q, \tau, p, f, q') \arrow[d, "\text{reads}"] \\
\mu(\tau) & & \omega & \tau = (q, \omega, p, f, q')
\end{tikzcd}$$

Weight separation for weighted automata with storage

Herrmann and Vogler (2015, Theorem 6)

$$k\text{-TSL}_{r,\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-TSL}_r)$$

Construction sketch.

$$\begin{tikzcd}
 & \mathcal{M}\colon \tau = (q, \omega, p, f, q') \arrow[ld, "\mu"'] \arrow[rd, "\text{reads}"] \arrow[rr, "\text{construction}"] & & \mathcal{M}'\colon (q, \tau, p, f, q') \arrow[d, "\text{reads}"] \\
 & & & \tau = (q, \omega, p, f, q') \arrow[d, "h"] \\
\mu(\tau) & & \omega & \mu(\tau).\omega
\end{tikzcd}$$

Original theorem is more general:

- *unital valuation monoid* instead of complete commutative strong bimonoid
- automata with arbitrary storage instead of tree stack automata

Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

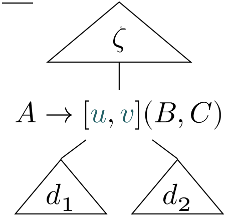
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

Construction sketch.

$\underline{G}$:



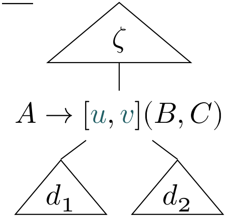
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

Construction sketch.

$\underline{G}$:



ζ

$A \to [u, v](B, C)$

d_1 d_2

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

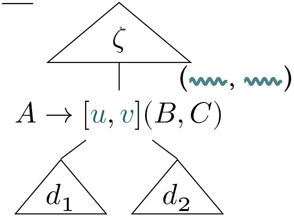
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

Construction sketch.

$\underline{G}$:



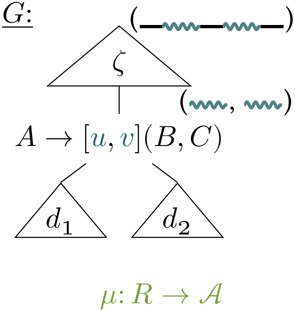
$\mu\colon R \to \mathcal{A}$

Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

Construction sketch.



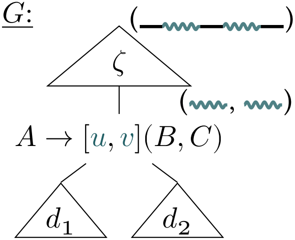
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

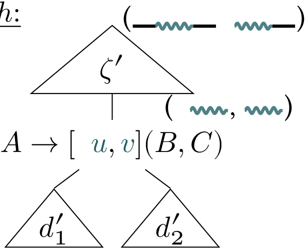
Construction sketch.

$\underline{G}$:



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

$\underline{G'}$ and $\underline{h}$:



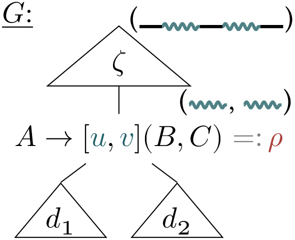
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

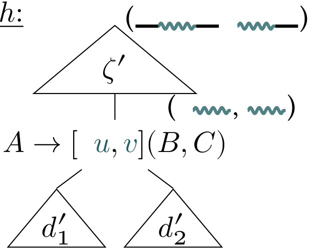
Construction sketch.

$\underline{G}$:



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

$\underline{G' \text{ and } h}$:



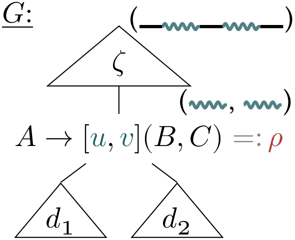
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

Construction sketch.

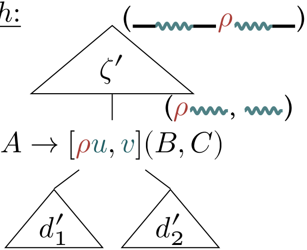
$\underline{G}$:



$A \to [u, v](B, C) =: \rho$

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

$\underline{G' \text{ and } h}$:



$A \to [\rho u, v](B, C)$

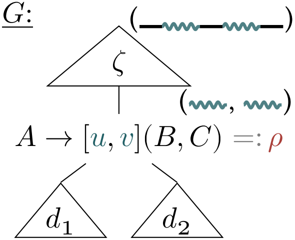
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

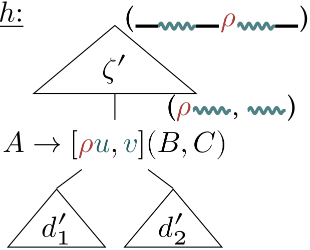
Construction sketch.

$\underline{G}$:



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

$\underline{G'}$ and h:



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

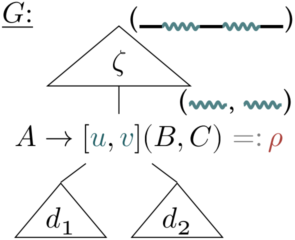
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

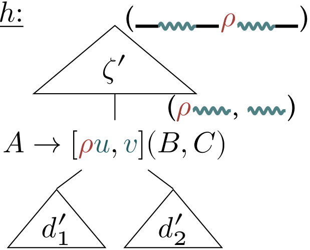
Construction sketch.

$\underline{G}$:



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

$\underline{G'}$ and $\underline{h}$:



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

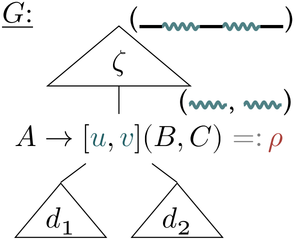
Weight separation for weighted MCFGs

Denkinger (2015, Lemma 3)

$$k\text{-MCFL}_{\mathcal{A}} = \alpha\text{HOM}_{\mathcal{A}}(k\text{-MCFL})$$

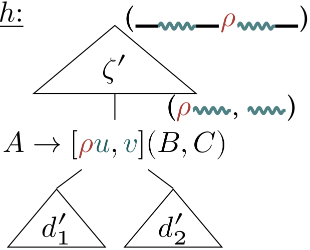
Construction sketch.

$\underline{G:}$



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

$\underline{G' \text{ and } h:}$



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