Coarse-to-fine recognition for weighted tree-stack automata

Max Korn

27. Oktober 2017

Motivation

- Problem: Parsing with complicated grammars and recognition with complicated automata are time intensive

Motivation

- Problem: Parsing with complicated grammars and recognition with complicated automata are time intensive

- Example: multiple-context-free grammars and tree-stack automata with restricted fanout k have a complexity of $\mathcal{O}(n^{3k})$

Motivation

- Problem: Parsing with complicated grammars and recognition with complicated automata are time intensive
- Example: multiple-context-free grammars and tree-stack automata with restricted fanout k have a complexity of $\mathcal{O}(n^{3k})$
- Solution: use less complex grammar/ automaton

Outline

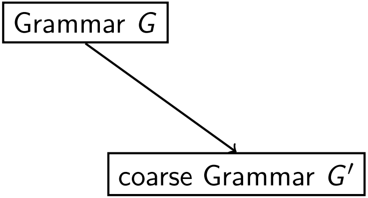
Outline

Grammar Based Coarse-to-Fine Parsing

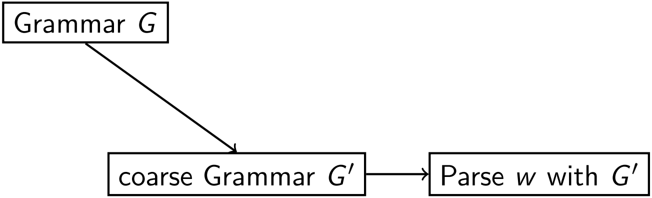
Grammar Based Coarse-to-Fine Parsing

$\boxed{\text{Grammar } G}$

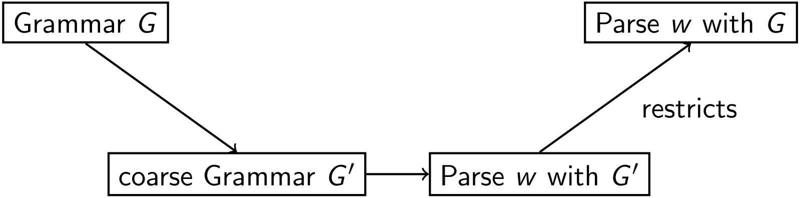
Grammar Based Coarse-to-Fine Parsing



Grammar Based Coarse-to-Fine Parsing



Grammar Based Coarse-to-Fine Parsing



Data-Storage

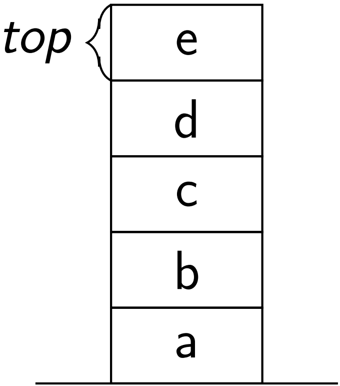
A tuple $S = (C, P, R, c_i)$ with

Data-Storage

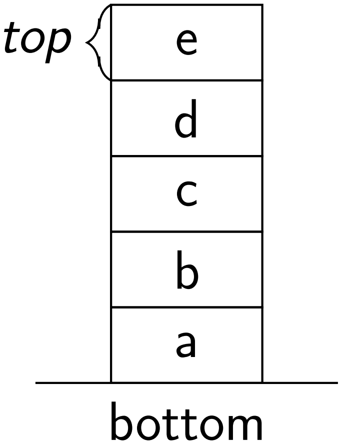
A tuple $S = (C, P, R, c_i)$ with

- set C (of configurations)
- set P (of predicates) with $P \subseteq P(C)$
- set R (of instructions) with $R \subseteq P(C \times C)$
- initial configuration $c_i \in C$

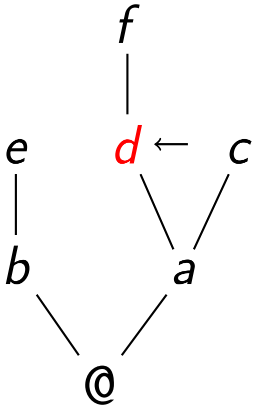
Push-Down



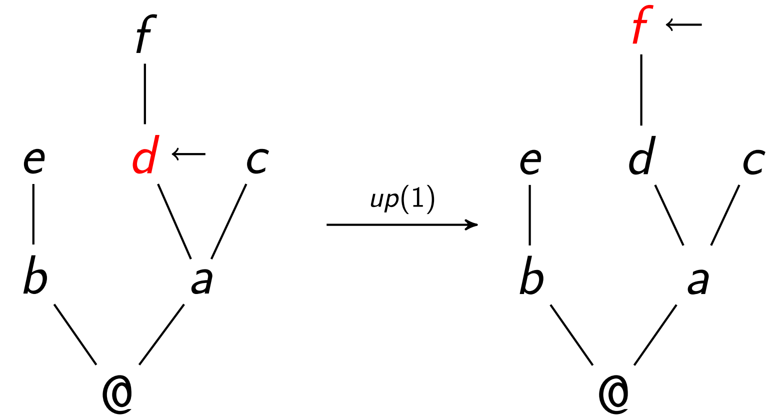
Push-Down



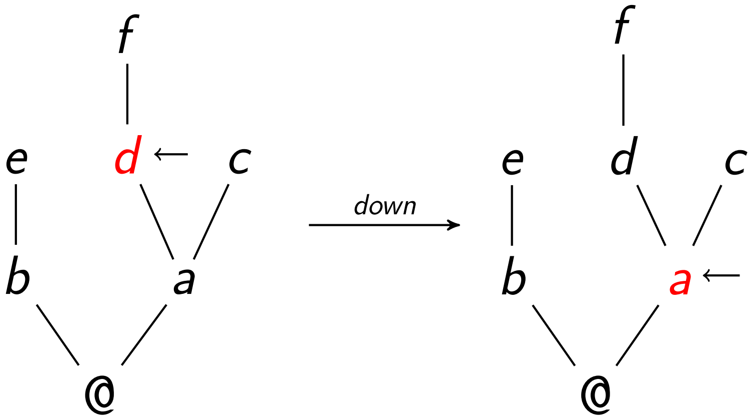
Tree-Stack



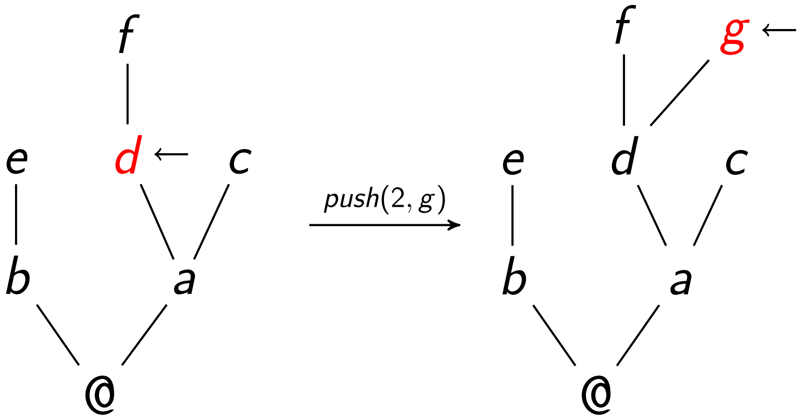
Tree-Stack



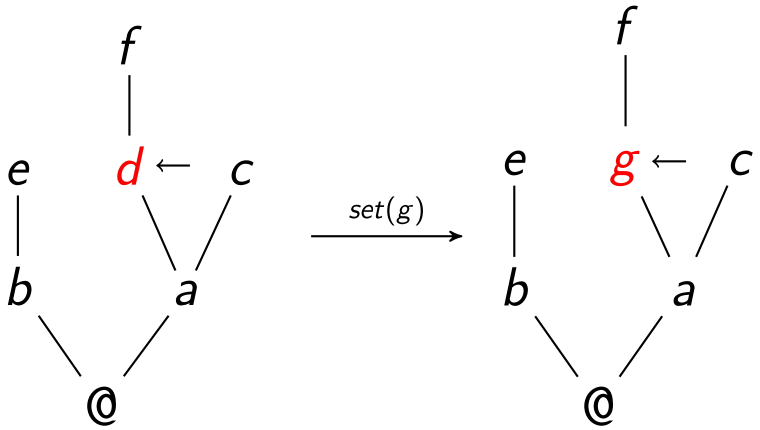
Tree-Stack



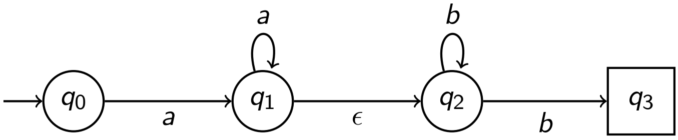
Tree-Stack



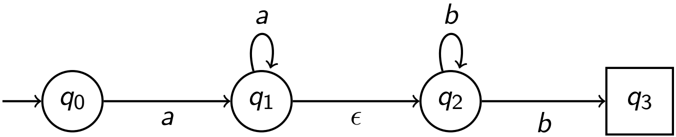
Tree-Stack



Automata

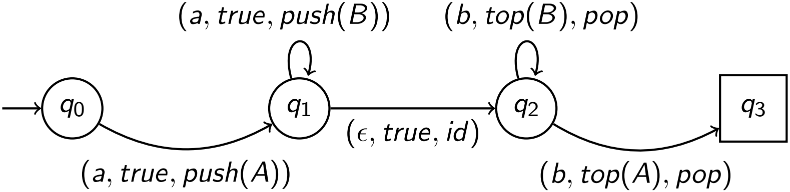


Automata

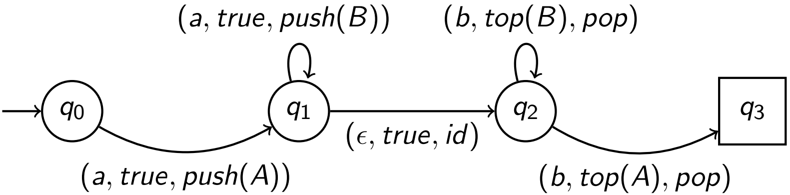


$$L(\mathcal{M}) = \{a^n b^m \mid n, m \geq 1\}$$

Automata with Storage

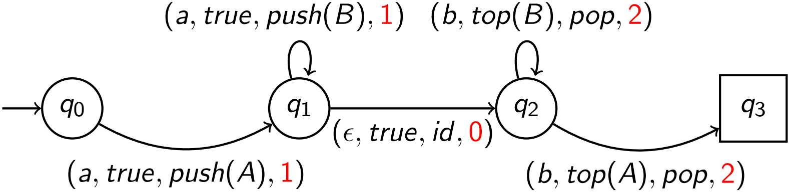


Automata with Storage



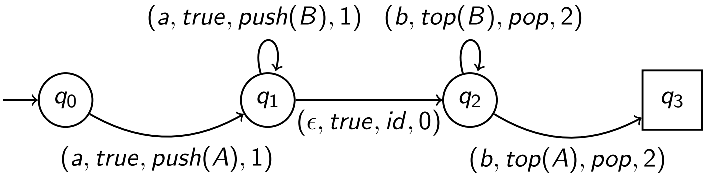
$$L(\mathcal{M}_S) = \{a^n b^n \mid n \geq 1\}$$

Automata with Storage

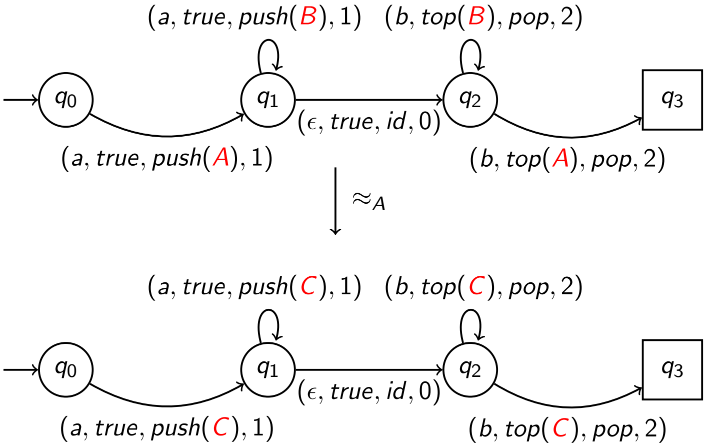


$$L(\mathcal{M}_S) = \{a^n b^n \mid n \geq 1\}$$

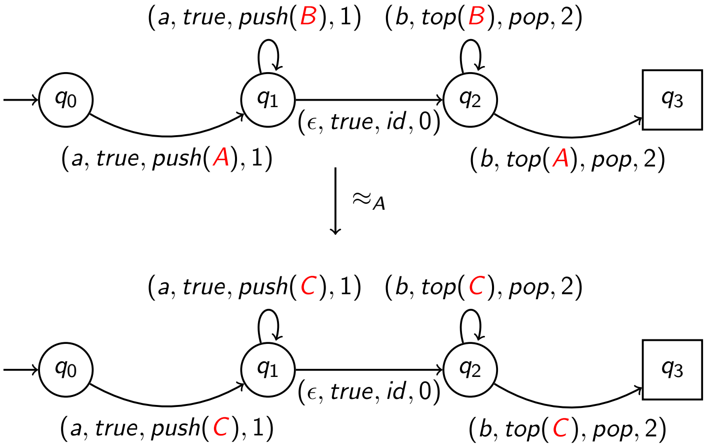
Approximation



Approximation



Approximation



$$L(\mathcal{M}_{\approx_A}) = \{a^n b^m \mid n \geq m \geq 1\}$$

Outline

We use three strategies to approximate the initial automaton:

We use three strategies to approximate the initial automaton:

- Ignoring tree-structures inspired by Burden and Ljunglöf [1] and Cranenburgh [3] (*TTS*)

We use three strategies to approximate the initial automaton:

- Ignoring tree-structures inspired by Burden and Ljunglöf [1] and Cranenburgh [3] (*TTS*)
- Relabelling to equivalence classes of stack symbols by Charniak et al. [2] (*RLB*)

We use three strategies to approximate the initial automaton:

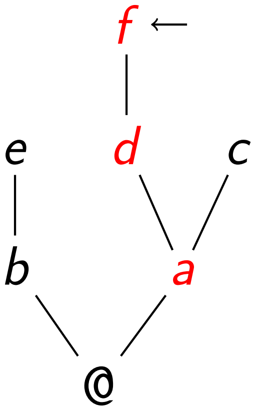
- Ignoring tree-structures inspired by Burden and Ljunglöf [1] and Cranenburgh [3] (*TTS*)
- Relabelling to equivalence classes of stack symbols by Charniak et al. [2] (*RLB*)
- Reducing the amount of push-down configurations to a finite number by Nederhof [5] (*PTK*)

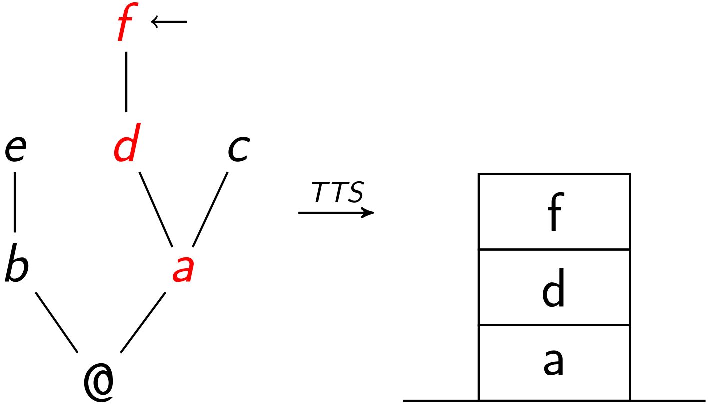
Ignore Tree-Structure

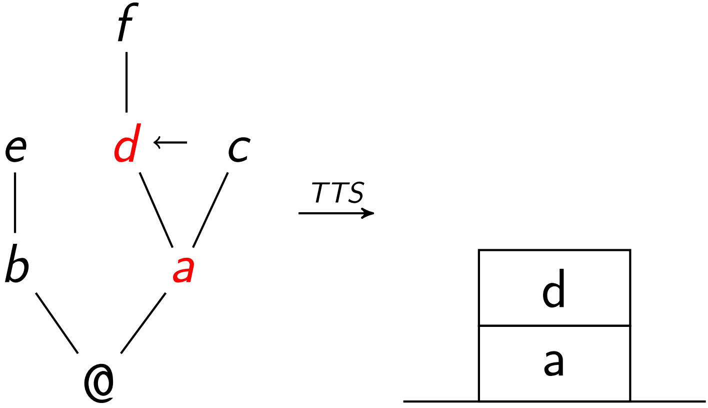
$\boxed{\text{Initial } \mathcal{M}}$

Ignore Tree-Structure

$$\begin{tikzcd} \boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] \\ \boxed{\mathcal{M}_{PD}} \end{tikzcd}$$


f
e
d
c
b
a
@


f ←
e
d
c
b
a
@
TTS
f
d
a

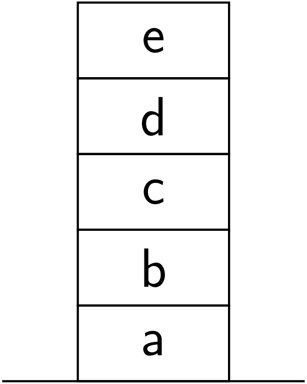

f
e
d
c
b
a
@
TTS
d
a

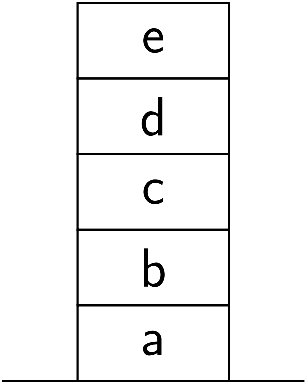
Equivalence-Classes of Stack symbols

$$\begin{tikzcd} \boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] \\ \boxed{\mathcal{M}_{PD}} \end{tikzcd}$$

Equivalence-Classes of Stack symbols

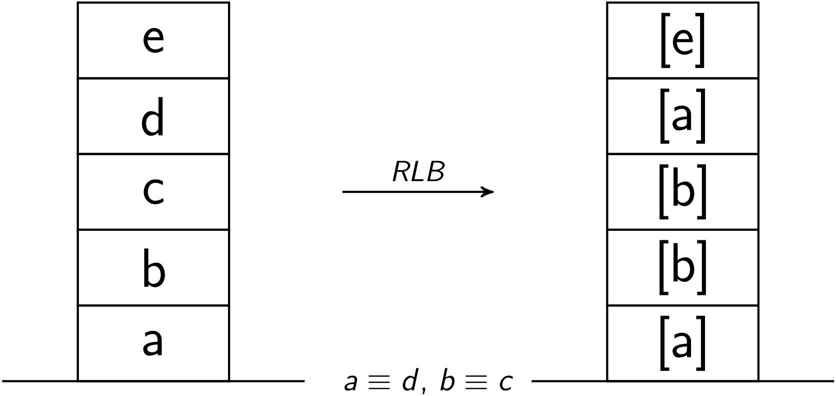
$$
\begin{tikzcd}
\boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] \\
\boxed{\mathcal{M}_{PD}} \arrow[d, "\approx_{RLB}"] \\
\boxed{\mathcal{M}_{Rel}}
\end{tikzcd}
$$


e
d
c
b
a



$\xrightarrow{RLB}$

$a \equiv d,\ b \equiv c$

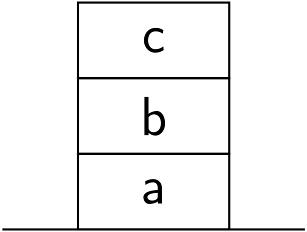

e
d
c
b
a
RLB
[e]
[a]
[b]
[b]
[a]
$a \equiv d$, $b \equiv c$

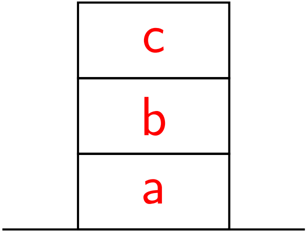
Limit Push-Down Height

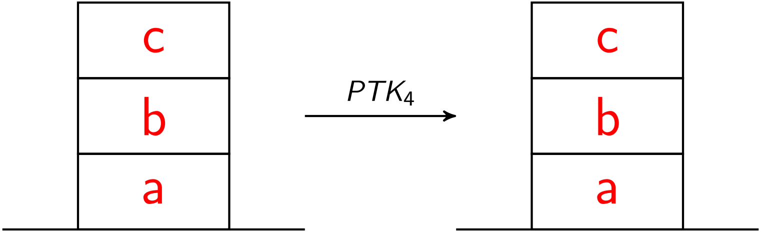
$$\begin{tikzcd}
\boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] \\
\boxed{\mathcal{M}_{PD}} \arrow[d, "\approx_{RLB}"] \\
\boxed{\mathcal{M}_{Rel}}
\end{tikzcd}$$

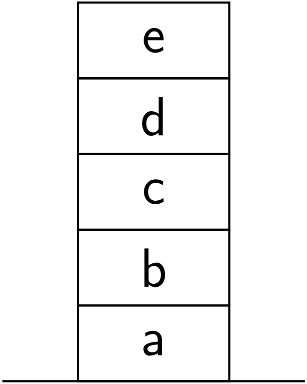
Limit Push-Down Height

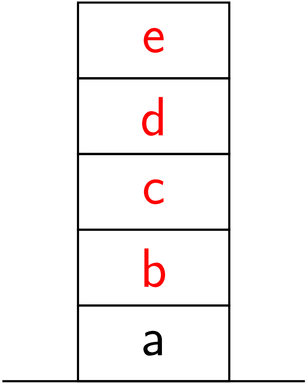
$$
\begin{tikzcd}
\boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] \\
\boxed{\mathcal{M}_{PD}} \arrow[d, "\approx_{RLB}"] \\
\boxed{\mathcal{M}_{Rel}} \arrow[d, "\approx_{PTK}"] \\
\boxed{\mathcal{M}_{NFA}}
\end{tikzcd}
$$

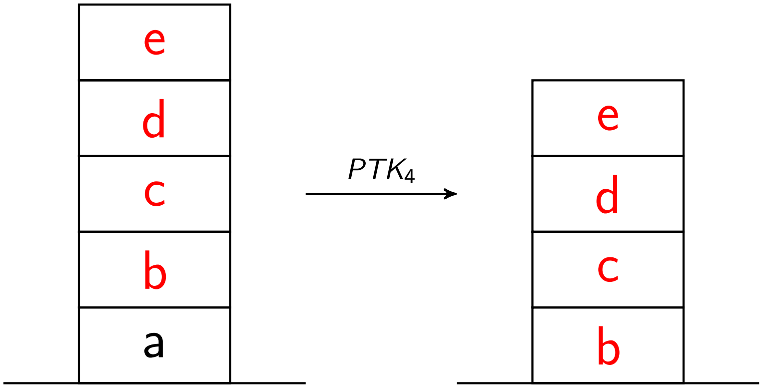

c
b
a


c
b
a


c
b
a
PTK_4
c
b
a

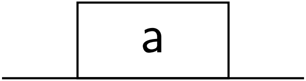

e
d
c
b
a


e
d
c
b
a


e
d
c
b
a
PTK_4
e
d
c
b

Special Case

pushdown

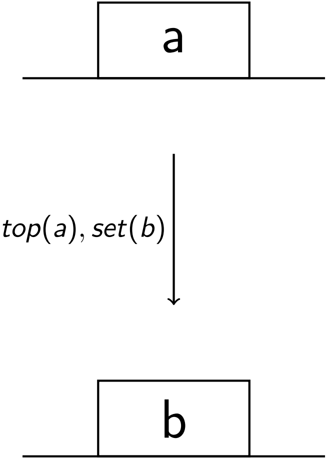


PTK_4(pushdown)

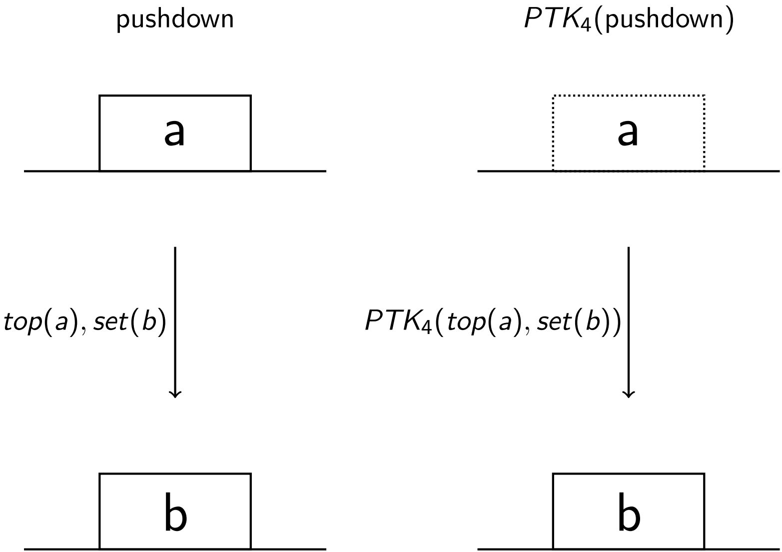
Special Case

pushdown

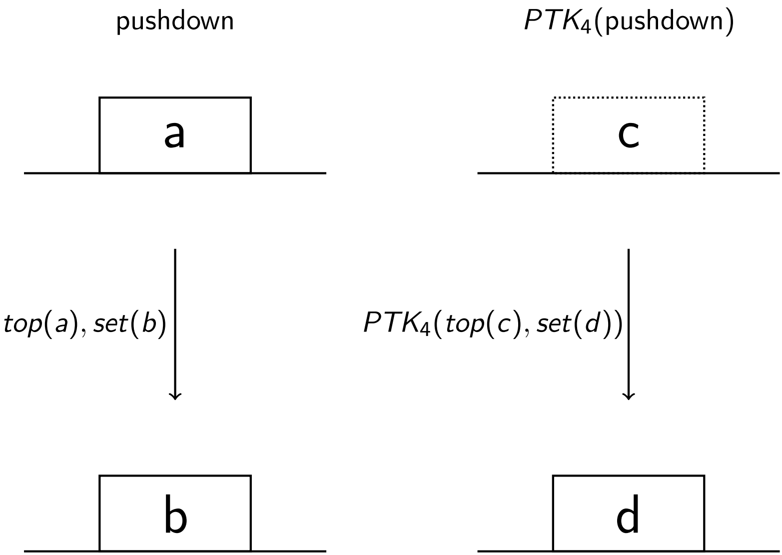
PTK_4(pushdown)



Special Case



Special Case



Outline

$$
\begin{array}{c}
\boxed{\text{Initial } \mathcal{M}} \\
\downarrow \approx_{TTS} \\
\boxed{\mathcal{M}_{PD}} \\
\downarrow \approx_{RLB} \\
\boxed{\mathcal{M}_{Rel}} \\
\downarrow \approx_{PTK} \\
\boxed{\mathcal{M}_{NFA}}
\end{array}
$$

$$
\begin{tikzcd}
\boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] & \\
\boxed{\mathcal{M}_{PD}} \arrow[d, "\approx_{RLB}"] & \\
\boxed{\mathcal{M}_{Rel}} \arrow[d, "\approx_{PTK}"] & \\
\boxed{\mathcal{M}_{NFA}} \arrow[r] & \boxed{\text{Recognise } w \text{ with } \mathcal{M}_{NFA}}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\boxed{\text{Initial } \mathcal{M}} \arrow[d, "\approx_{TTS}"] & & \\
\boxed{\mathcal{M}_{PD}} \arrow[d, "\approx_{RLB}"] & & \\
\boxed{\mathcal{M}_{Rel}} \arrow[d, "\approx_{PTK}"] & & \\
\boxed{\mathcal{M}_{NFA}} \arrow[r] & \boxed{\text{Recognise } w \text{ with } \mathcal{M}_{NFA}} \arrow[dr, "\text{No run found}"'] & \\
& & \boxed{\text{Does not recognise}}
\end{tikzcd}
$$

Algorithm by Denkinger [4]

Input: proper total approximation strategy A,
(S, Σ, K)-Automaton $\mathcal{M}$, $n \in \mathbb{N}$, word $w \in \Sigma^*$ and part. order $(\leq) \subseteq K \times K$

Output: some set of n-best runs of $\mathcal{M}$ on w

Algorithm by Denkinger [4]

Input: proper total approximation strategy A,
(S, Σ, K)-Automaton $\mathcal{M}$, $n \in \mathbb{N}$, word $w \in \Sigma^*$ and part. order $(\leq) \subseteq K \times K$

Output: some set of n-best runs of $\mathcal{M}$ on w

$\mathcal{M}' \leftarrow \approx_A (\mathcal{M})$

$P_f \leftarrow \emptyset$

$P_c \leftarrow R_{\mathcal{M}'}(w)$

return P_f

Algorithm by Denkinger [4]

Input: proper total approximation strategy A,
(S, Σ, K)-Automaton $\mathcal{M}$, $n \in \mathbb{N}$, word $w \in \Sigma^*$ and part. order
$(\leq) \subseteq K \times K$

Output: some set of n-best runs of $\mathcal{M}$ on w

$\mathcal{M}' \leftarrow \approx_A (\mathcal{M})$
$P_f \leftarrow \emptyset$
$P_c \leftarrow R_{\mathcal{M}'}(w)$
while **do**
    $\theta \leftarrow$ smallest element of P_c
    $P_c \leftarrow P_c \setminus \{\theta\}$

return P_f

Algorithm by Denkinger [4]

Input: proper total approximation strategy A,
(S, Σ, K)-Automaton $\mathcal{M}$, $n \in \mathbb{N}$, word $w \in \Sigma^*$ and part. order $(\leq) \subseteq K \times K$

Output: some set of n-best runs of $\mathcal{M}$ on w

$\mathcal{M}' \leftarrow \approx_A (\mathcal{M})$
$P_f \leftarrow \emptyset$
$P_c \leftarrow R_{\mathcal{M}'}(w)$
while **do**
  $\theta \leftarrow$ smallest element of P_c
  $P_c \leftarrow P_c \setminus \{\theta\}$
  for $\theta' \in \approx_A^{-1} (\theta)$ **do**
    if $\theta' \in R_{\mathcal{M}}$ **then** $P_f \leftarrow P_f \cup \{\theta'\}$
return P_f

Algorithm by Denkinger [4]

Input: proper total approximation strategy A,
(S, Σ, K)-Automaton $\mathcal{M}$, $n \in \mathbb{N}$, word $w \in \Sigma^*$ and part. order $(\leq) \subseteq K \times K$

Output: some set of n-best runs of $\mathcal{M}$ on w

$\mathcal{M}' \leftarrow \approx_A (\mathcal{M})$
$P_f \leftarrow \emptyset$
$P_c \leftarrow R_{\mathcal{M}'}(w)$
while $|P_f| < n$ or $max_{\theta \in P_f} wt(\theta) > min_{\theta' \in P_c} wt(\approx_A^{-1} (\theta'))$ **do**
    $\theta \leftarrow$ smallest element of P_c
    $P_c \leftarrow P_c \setminus \{\theta\}$
    for $\theta' \in \approx_A^{-1} (\theta)$ **do**
        if $\theta' \in R_{\mathcal{M}}$ **then** $P_f \leftarrow P_f \cup \{\theta'\}$
return P_f

Algorithm for multiple layers

$$\mathcal{M} \xrightarrow{\approx_{A_1}} \mathcal{M}_1 \xrightarrow{\approx_{A_2}} \mathcal{M}_2 \xrightarrow{\approx_{A_3}} \dots \xrightarrow{\approx_{A_m}} \mathcal{M}_m$$

Algorithm for multiple layers

$$\mathcal{M} \xrightarrow{\approx_{A_1}} \mathcal{M}_1 \xrightarrow{\approx_{A_2}} \mathcal{M}_2 \xrightarrow{\approx_{A_3}} \dots \xrightarrow{\approx_{A_m}} \mathcal{M}_m$$

Input: (S, Σ, K)-Automaton $\mathcal{M}$, $n \in \mathbb{N}$, word $w \in \Sigma^*$, part. order
$(\leq) \subseteq K \times K$,
proper total approximation strategy A_1,
proper total approximation strategy A_2,
proper total approximation strategy A_3,
$\dots$,
proper total approximation strategy A_m,
Output: some set of n-best runs of $\mathcal{M}$ on w

Algorithm for multiple layers

$\mathcal{M}_1 \leftarrow \approx_{A_1} (\mathcal{M})$

$\mathcal{M}_2 \leftarrow \approx_{A_2} (\mathcal{M}_1)$

$\dots$

$\mathcal{M}_m \leftarrow \approx_{A_m} (\mathcal{M}_{m-1})$

$P_f \leftarrow \emptyset$

$P_m \leftarrow R_{\mathcal{M}_m}(w)$

return P_f

Algorithm for multiple layers

$\mathcal{M}_1 \leftarrow \approx_{A_1} (\mathcal{M})$
$\mathcal{M}_2 \leftarrow \approx_{A_2} (\mathcal{M}_1)$
$\dots$
$\mathcal{M}_m \leftarrow \approx_{A_m} (\mathcal{M}_{m-1})$
$P_f \leftarrow \emptyset$
$P_m \leftarrow R_{\mathcal{M}_m}(w)$
while $|P_f| < n$ or $\max_{\theta \in P_f} wt(\theta) > \min_{\theta' \in P_m} wt(\approx_A^{-1} (\theta'))$ **do**
    $\theta_m \leftarrow$ smallest element of P_m
    $P_m \leftarrow P_m \setminus \{\theta_m\}$

return P_f

Algorithm for multiple layers

$\mathcal{M}_1 \leftarrow \approx_{A_1} (\mathcal{M})$
$\mathcal{M}_2 \leftarrow \approx_{A_2} (\mathcal{M}_1)$
$\dots$
$\mathcal{M}_m \leftarrow \approx_{A_m} (\mathcal{M}_{m-1})$
$P_f \leftarrow \emptyset$
$P_m \leftarrow R_{\mathcal{M}_m}(w)$
while $|P_f| < n$ or $\max_{\theta \in P_f} wt(\theta) > \min_{\theta' \in P_m} wt(\approx_A^{-1} (\theta'))$ **do**
  $\theta_m \leftarrow$ smallest element of P_m
  $P_m \leftarrow P_m \setminus \{\theta_m\}$
  for $\theta_{m-1} \in \approx_{A_m}^{-1} (\theta_m)$ **do**
    if $\theta_{m-1} \in R_{\mathcal{M}_{m-1}}$ **then**

return P_f

Algorithm for multiple layers

$\mathcal{M}_1 \leftarrow \approx_{A_1} (\mathcal{M})$

$\mathcal{M}_2 \leftarrow \approx_{A_2} (\mathcal{M}_1)$

$\dots$

$\mathcal{M}_m \leftarrow \approx_{A_m} (\mathcal{M}_{m-1})$

$P_f \leftarrow \emptyset$

$P_m \leftarrow R_{\mathcal{M}_m}(w)$

while $|P_f| < n$ or $\max_{\theta \in P_f} wt(\theta) > \min_{\theta' \in P_m} wt(\approx_A^{-1} (\theta'))$ **do**

  $\theta_m \leftarrow$ smallest element of P_m

  $P_m \leftarrow P_m \setminus \{\theta_m\}$

  for $\theta_{m-1} \in \approx_{A_m}^{-1} (\theta_m)$ **do**

    if $\theta_{m-1} \in R_{\mathcal{M}_{m-1}}$ **then**

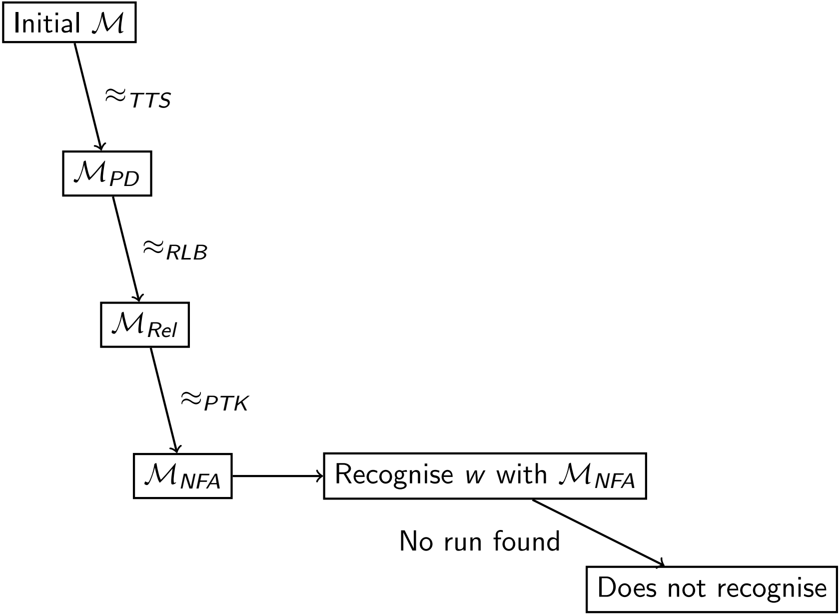
      for $\theta_{m-2} \in \approx_{A_{m-1}}^{-1} (\theta_{m-1})$ **do**

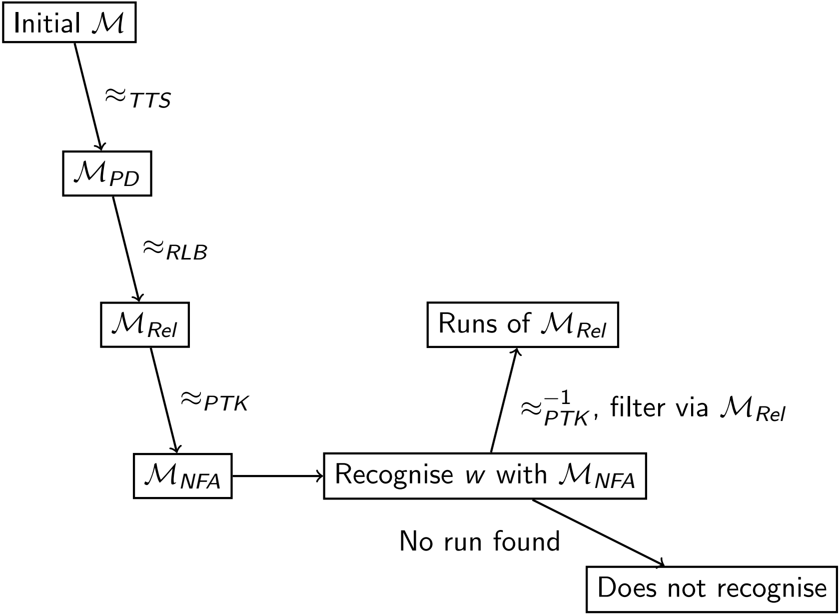
        if $\theta_{m-2} \in R_{\mathcal{M}_{m-2}}$ **then**

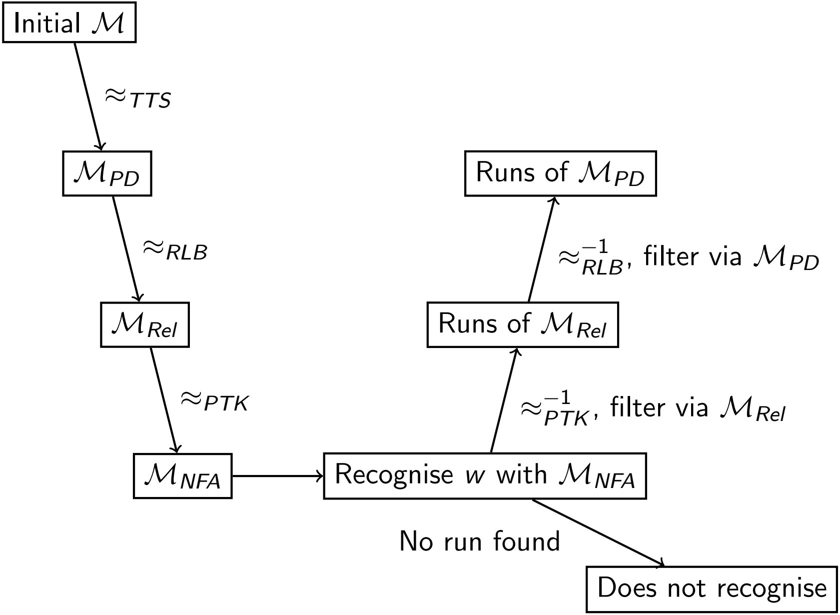
return P_f

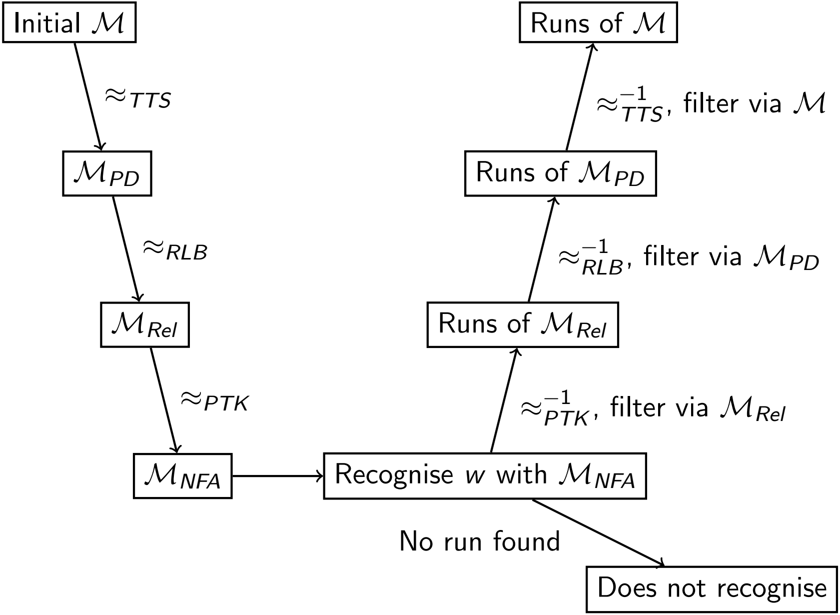
Algorithm for multiple layers

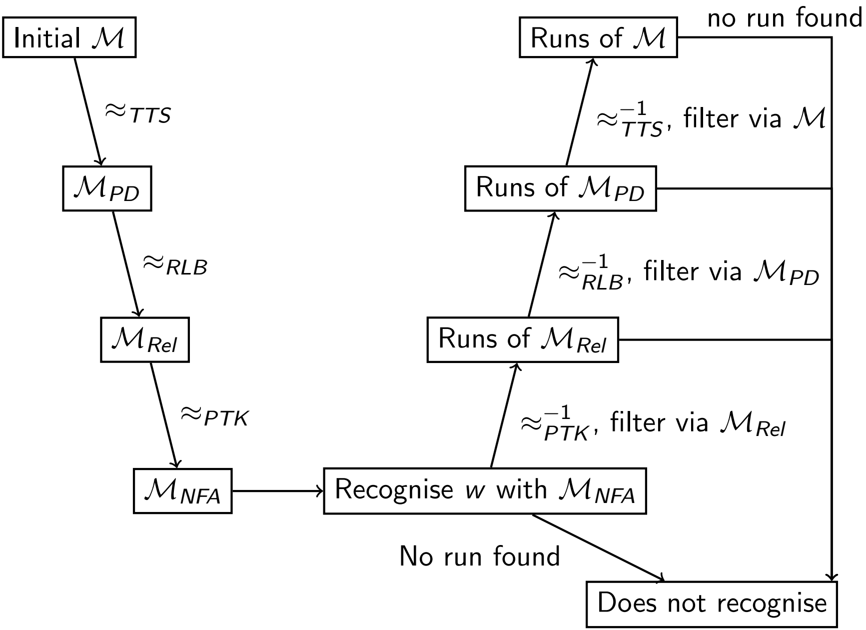
$\mathcal{M}_1 \leftarrow \approx_{A_1} (\mathcal{M})$
$\mathcal{M}_2 \leftarrow \approx_{A_2} (\mathcal{M}_1)$
$\dots$
$\mathcal{M}_m \leftarrow \approx_{A_m} (\mathcal{M}_{m-1})$
$P_f \leftarrow \emptyset$
$P_m \leftarrow R_{\mathcal{M}_m}(w)$
while $|P_f| < n$ or $\max_{\theta \in P_f} wt(\theta) > \min_{\theta' \in P_m} wt(\approx_A^{-1} (\theta'))$ **do**
    $\theta_m \leftarrow$ smallest element of P_m
    $P_m \leftarrow P_m \setminus \{\theta_m\}$
    for $\theta_{m-1} \in \approx_{A_m}^{-1} (\theta_m)$ **do**
        if $\theta_{m-1} \in R_{\mathcal{M}_{m-1}}$ **then**
            for $\theta_{m-2} \in \approx_{A_{m-1}}^{-1} (\theta_{m-1})$ **do**
                if $\theta_{m-2} \in R_{\mathcal{M}_{m-2}}$ **then**
                    $\dots$
                    for $\theta_0 \in \approx_{A_1}^{-1} (\theta_1)$ **do**
                        if $\theta_0 \in R_{\mathcal{M}}$ **then** $P_f \leftarrow P_f \cup \{\theta_0\}$
return P_f


Initial $\mathcal{M}$
$\approx_{TTS}$
$\mathcal{M}_{PD}$
$\approx_{RLB}$
$\mathcal{M}_{Rel}$
$\approx_{PTK}$
$\mathcal{M}_{NFA}$
Recognise w with $\mathcal{M}_{NFA}$
No run found
Does not recognise


Initial $\mathcal{M}$
$\approx_{TTS}$
$\mathcal{M}_{PD}$
$\approx_{RLB}$
$\mathcal{M}_{Rel}$
$\approx_{PTK}$
$\mathcal{M}_{NFA}$
Recognise w with $\mathcal{M}_{NFA}$
$\approx_{PTK}^{-1}$, filter via $\mathcal{M}_{Rel}$
Runs of $\mathcal{M}_{Rel}$
No run found
Does not recognise


Initial $\mathcal{M}$
$\approx_{TTS}$
$\mathcal{M}_{PD}$
$\approx_{RLB}$
$\mathcal{M}_{Rel}$
$\approx_{PTK}$
$\mathcal{M}_{NFA}$
Recognise w with $\mathcal{M}_{NFA}$
$\approx_{PTK}^{-1}$, filter via $\mathcal{M}_{Rel}$
Runs of $\mathcal{M}_{Rel}$
$\approx_{RLB}^{-1}$, filter via $\mathcal{M}_{PD}$
Runs of $\mathcal{M}_{PD}$
No run found
Does not recognise


Initial $\mathcal{M}$
$\approx_{TTS}$
$\mathcal{M}_{PD}$
$\approx_{RLB}$
$\mathcal{M}_{Rel}$
$\approx_{PTK}$
$\mathcal{M}_{NFA}$
Recognise w with $\mathcal{M}_{NFA}$
No run found
Does not recognise
$\approx_{PTK}^{-1}$, filter via $\mathcal{M}_{Rel}$
Runs of $\mathcal{M}_{Rel}$
$\approx_{RLB}^{-1}$, filter via $\mathcal{M}_{PD}$
Runs of $\mathcal{M}_{PD}$
$\approx_{TTS}^{-1}$, filter via $\mathcal{M}$
Runs of $\mathcal{M}$



Initial $\mathcal{M}$
$\approx_{TTS}$
$\mathcal{M}_{PD}$
$\approx_{RLB}$
$\mathcal{M}_{Rel}$
$\approx_{PTK}$
$\mathcal{M}_{NFA}$
Recognise w with $\mathcal{M}_{NFA}$
No run found
Does not recognise
$\approx_{PTK}^{-1}$, filter via $\mathcal{M}_{Rel}$
Runs of $\mathcal{M}_{Rel}$
$\approx_{RLB}^{-1}$, filter via $\mathcal{M}_{PD}$
Runs of $\mathcal{M}_{PD}$
$\approx_{TTS}^{-1}$, filter via $\mathcal{M}$
Runs of $\mathcal{M}$
no run found

Outline

Implementation[1]

[1] using *rustomata* https://github.com/tud-fop/rustomata

Implementation[1]

Preprocessing: $\texttt{app0} \xrightarrow{\texttt{tts}} \texttt{app1} \xrightarrow{\texttt{rlb}} \texttt{app2} \xrightarrow{\texttt{ptk}} \texttt{app3}$

[1] using *rustomata* `https://github.com/tud-fop/rustomata`

Implementation[1]

Preprocessing: $\mathsf{app0} \xrightarrow{\texttt{tts}} \mathsf{app1} \xrightarrow{\texttt{rlb}} \mathsf{app2} \xrightarrow{\texttt{ptk}} \mathsf{app3}$

```
for run1 in app3.recognise(word).take(n) {
  let trans_runs1 = ctf_level(run1, &ptk, &app2);
  for run2 in trans_runs1 {
    let trans_runs2 = ctf_level(run2, &rlb, &app1);
    for run3 in trans_runs2 {
      let trans_runs3 = ctf_level(run3, &tts, &app0);
      for run4 in trans_runs3 {
        println!("{:?}", run4);
      }
    }
  }
}
```

[1] using *rustomata* `https://github.com/tud-fop/rustomata`

Outline

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]

[2] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html

[3] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*

[2] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html

[3] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]

[2] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html

[3] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]
- *PTK* heights of 5, 10, 15 and 20

[2] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc`

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]
- *PTK* heights of 5, 10, 15 and 20
- recognising three sentences contained in the corresponding corpus

---

[2] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html

[3] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]
- *PTK* heights of 5, 10, 15 and 20
- recognising three sentences contained in the corresponding corpus

$$\mathcal{M} \xrightarrow[TTS]{} \mathcal{M}_{PD} \xrightarrow[RLB]{} \mathcal{M}_{Rel} \xrightarrow[PTK]{} \mathcal{M}_{NFA}$$

[2] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc`

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]
- *PTK* heights of 5, 10, 15 and 20
- recognising three sentences contained in the corresponding corpus

1 layer

$$\mathcal{M} \xrightarrow[TTS]{} \mathcal{M}_{PD} \xrightarrow[RLB]{} \mathcal{M}_{Rel} \xrightarrow[PTK]{} \mathcal{M}_{NFA}$$

[2] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc`

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]
- *PTK* heights of 5, 10, 15 and 20
- recognising three sentences contained in the corresponding corpus

2 layers

$$\mathcal{M} \xrightarrow[TTS]{} \mathcal{M}_{PD} \xrightarrow[RLB]{} \mathcal{M}_{Rel} \xrightarrow[PTK]{} \mathcal{M}_{NFA}$$

---

[2] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc`

Experiments

- Grammars created by using the first 5, 10, 15 and 20 sentences of the NEGRA corpus[2]
- Grammars converted into Automata by *rustomata*
- 2 equivalence classes, one using fanout and the other using tag-grouping[3]
- *PTK* heights of 5, 10, 15 and 20
- recognising three sentences contained in the corresponding corpus

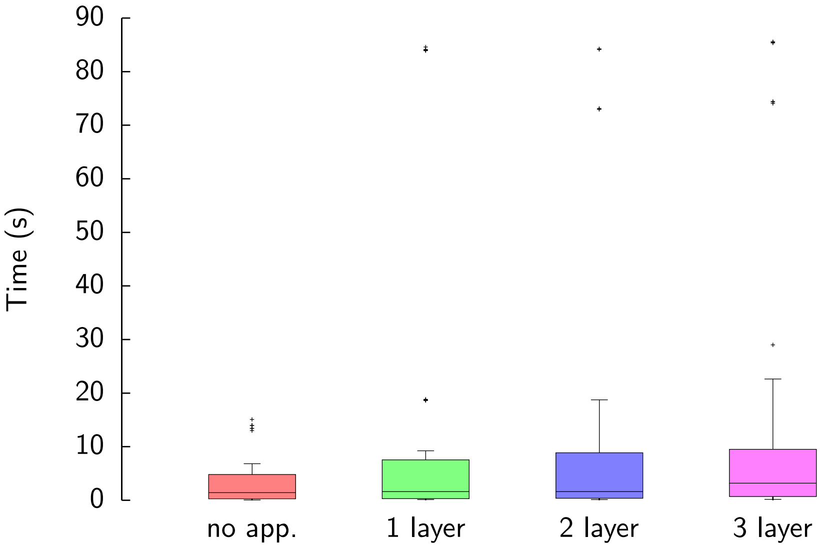
3 layers

$$\mathcal{M} \xrightarrow[TTS]{} \mathcal{M}_{PD} \xrightarrow[RLB]{} \mathcal{M}_{Rel} \xrightarrow[PTK]{} \mathcal{M}_{NFA}$$

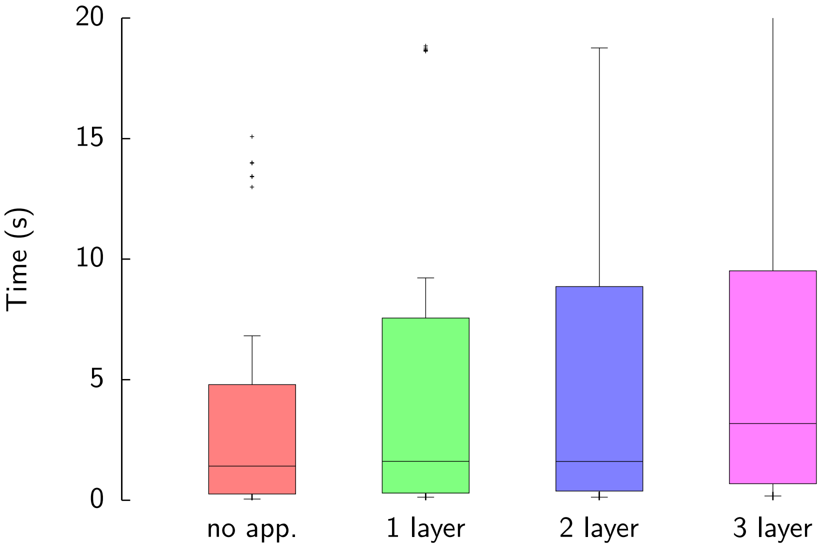
[2] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html

[3] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc

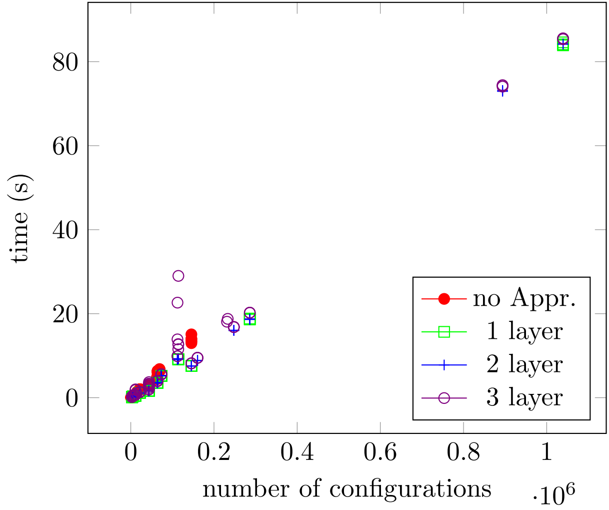
Results



Results

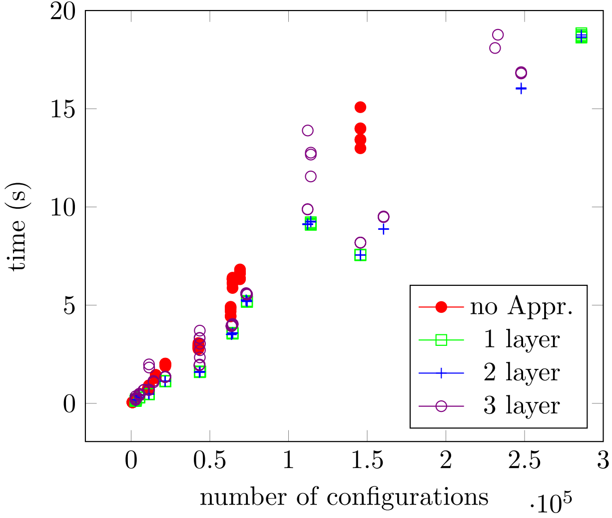


Results



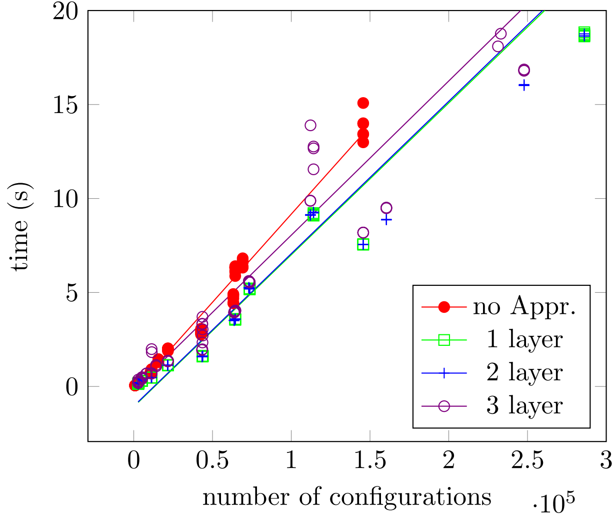
time (s)
80
60
40
20
0
0
0.2
0.4
0.6
0.8
1
number of configurations
$\cdot 10^6$
no Appr.
1 layer
2 layer
3 layer

Results



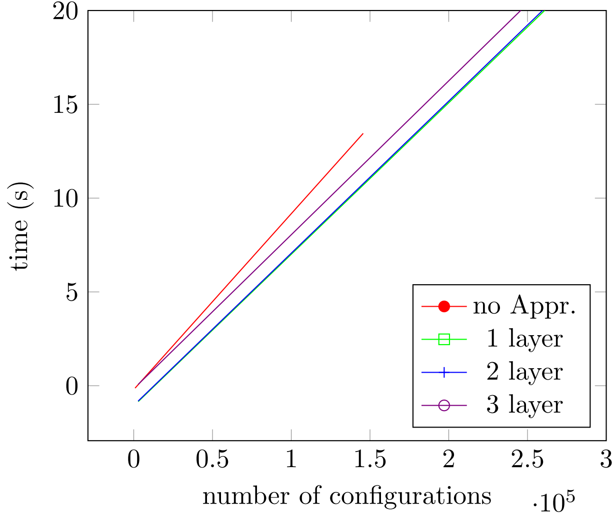
20
15
10
5
0
time (s)
0
0.5
1
1.5
2
2.5
3
number of configurations
$\cdot 10^5$
no Appr.
1 layer
2 layer
3 layer

Results



time (s)
20
15
10
5
0
0
0.5
1
1.5
2
2.5
3
number of configurations
$\cdot 10^5$
no Appr.
1 layer
2 layer
3 layer

Results



20
15
10
5
0
time (s)
0
0.5
1
1.5
2
2.5
3
number of configurations
$\cdot 10^5$
no Appr.
1 layer
2 layer
3 layer

References

[1] Håkan Burden and Peter Ljunglöf. "Parsing Linear Context-Free Rewriting Systems". In: *Proceedings of the Ninth IWPT* (2005), pp. 11–17.

[2] Eugene Charniak et al. "Multilevel coarse-to-fine PCFG parsing". In: *Proceedings of the HLT-NACL*. 2006.

[3] Andreas van Cranenburgh. "Efficient Parsing with Linear Context-Free Rewriting Systems". In: *Proceedings of the 13th Conference of the EACL.* (2012), pp. 460–470.

[4] Tobias Denkinger. "Approximation of Weighted Automata with Storage". In: *Proceedings Eighth International Symposium on GandALF*. 2017, pp. 91–105.

[5] Mark-Jan Nederhof. "Regular approximations of CFLs: A grammatical view". In: *Proceedings of the IWPT* (1997), pp. 159–170.

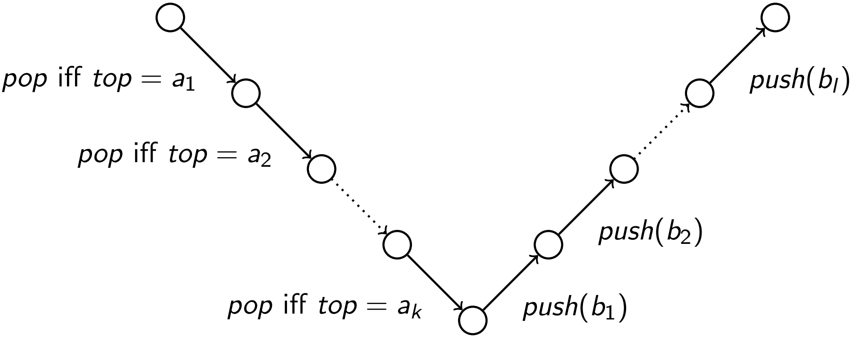
Stateless Push-Down Automata

Stateless Push-Down Automata

$$\text{replace}(\underbrace{a_1 a_2 a_3 \ldots a_k}_{\text{current symbols}}, \underbrace{b_1 b_2 b_3 \ldots b_l}_{\text{new symbols}})$$

Stateless Push-Down Automata

$$\text{replace}(\underbrace{a_1 a_2 a_3 \ldots a_k}_{\text{current symbols}}, \underbrace{b_1 b_2 b_3 \ldots b_l}_{\text{new symbols}})$$



- *push*$(\gamma) =$ *replace*(ε, γ) for all $\gamma \in \Gamma$

- $\mathit{push}(\gamma) = \mathit{replace}(\varepsilon, \gamma)$ for all $\gamma \in \Gamma$
- $\mathit{pop}(\gamma) = \mathit{replace}(\gamma, \varepsilon)$ for all $\gamma \in \Gamma$

- *push*(γ) = *replace*(ε, γ) for all $\gamma \in \Gamma$
- *pop*(γ) = *replace*(γ, ε) for all $\gamma \in \Gamma$
- *id*(γ) = *replace*(γ, γ) for all $\gamma \in \Gamma$

- *push*$(\gamma) =$ *replace*(ε, γ) for all $\gamma \in \Gamma$
- *pop*$(\gamma) =$ *replace*(γ, ε) for all $\gamma \in \Gamma$
- *id*$(\gamma) =$ *replace*(γ, γ) for all $\gamma \in \Gamma$
- *id* $=$ *replace*$(\varepsilon, \varepsilon)$

Stateless Tree-Stack Automata

Every instruction is preceded and followed by a *set*(γ).

Stateless Tree-Stack Automata

Every instruction is preceded and followed by a *set*(γ).

State behaviour is encoded into stack-symbols.