# Master Defense: Implementation and evaluation of $k$-best Chomsky-Schützenberger parsing for weighted multiple context-free grammars

Thomas Ruprecht

March 21, 2018

# CS parsing for context-free grammars [Hul09]

**Theorem (Chomsky-Schützenberger theorem for context-free languages [CS63])**

*Let $L$ be a language. T.f.a.e.*

1. *$L$ is a context-free language, and*
2. *there are a regular language $R$, a Dyck language $D$ and a string homomorphism $h$ such that $L = h(R \cap D)$.*

# CS parsing for context-free grammars [Hul09]

**Theorem (Chomsky-Schützenberger theorem for context-free languages [CS63])**

*Let $L$ be a language. T.f.a.e.*

1. *$L$ is a context-free language, and*
2. *there are a regular language $R$, a Dyck language $D$ and a string homomorphism $h$ such that $L = h(R \cap D)$.*

**Input:** Grammar $G$, word $w$
**Output:** sequence $d_1 d_2 \ldots$ of ASTs for $w$ in $G$

construct $R$, $D$, $h$ with respect to $[\![G]\!]$
$R^w \leftarrow h^{-1}(w)$
$R^{local} \leftarrow R^w \cap R$
**return** $\big(\mathrm{enumerate} \; ; \; \mathrm{filter}(D) \; ; \; \mathrm{map}(\mathrm{toderiv})\big)(R^{local})$

# CS parsing for context-free grammars [Hul09]: example

- example grammar $G\colon S \to SS \mid a$

# CS parsing for context-free grammars [Hul09]: example

- example grammar $G$: $S \to SS \mid a$
- Dyck words are well-bracketed

$$(^1_{S\to SS}\ (^1_{S\to a}\langle_a\ \rangle_a\ )^1_{S\to A}\ )^1_{S\to SS}\ (^2_{S\to SS}\ (^1_{S\to a}\langle_a\ \rangle_a\ )^1_{S\to A}\ )^2_{S\to SS}$$

# CS parsing for context-free grammars [Hul09]: example

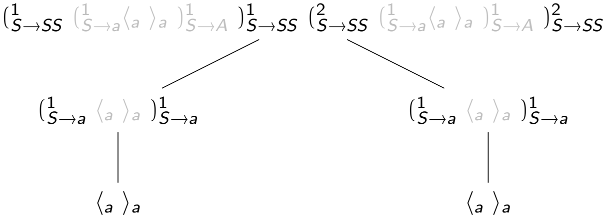
- example grammar $G\colon S \to SS \mid a$
- Dyck words are well-bracketed

$$(^1_{S\to SS}\ (^1_{S\to a}\langle_a\ \rangle_a\ )^1_{S\to A}\ )^1_{S\to SS}\ (^2_{S\to SS}\ (^1_{S\to a}\langle_a\ \rangle_a\ )^1_{S\to A}\ )^2_{S\to SS}$$

$$(^1_{S\to a}\ \langle_a\ \rangle_a\ )^1_{S\to a} \qquad\qquad (^1_{S\to a}\ \langle_a\ \rangle_a\ )^1_{S\to a}$$

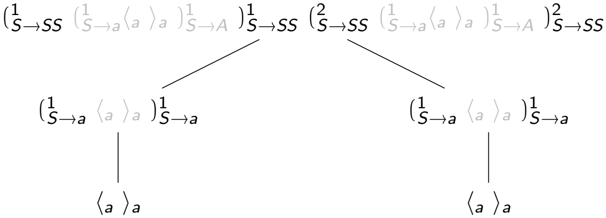
# CS parsing for context-free grammars [Hul09]: example

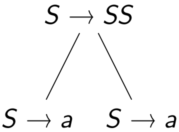
- example grammar $G\colon S \rightarrow SS \mid a$
- Dyck words are well-bracketed



# CS parsing for context-free grammars [Hul09]: example

- example grammar $G$: $S \to SS \mid a$
- Dyck words are well-bracketed
- structure of parenthesis hierarchy relates to abstract syntax tree



# outline

weighted multiple context-free grammars

CS parsing for WMCFG

implementation

evaluation results

conclusions and future work

# outline

weighted multiple context-free grammars

CS parsing for WMCFG

implementation

evaluation results

conclusions and future work

# weighted multiple context-free grammars

- rules of the form

$$A \longrightarrow [ax_1^1, cx_1^2](A)$$

# weighted multiple context-free grammars

- rules of the form

$$A \rightarrow [ax_1^1, cx_1^2](A)$$

- "CFG with tuples"

# weighted multiple context-free grammars

- rules of the form

$$A \longrightarrow [ax_1^1, cx_1^2](A)$$

- "CFG with tuples"
- composition is linear

# weighted multiple context-free grammars

- rules of the form

$$A \to [ax_1^1, cx_1^2](A)$$

- "CFG with tuples"
- composition is linear
- for each (weighted) MCFG, there is an equivalent (weighted) non-deleting MCFG [Sek+91; Den16]

# outline

weighted multiple context-free grammars

CS parsing for WMCFG

implementation

evaluation results

conclusions and future work

**Definition ($k$-best parsing problem for WMCFG)**

**Input** $W$-weighted MCFG $(G, \mu)$, word $w$ and total order $\trianglelefteq$ on $W$

**Output** sequence $d_1 \ldots d_k$ of ASTs for $w$ in $G$ s.t. $\mu(d_1) \trianglelefteq \ldots \trianglelefteq \mu(d_k)$ and $\nexists d \colon \mu(d_k) \ntrianglelefteq \mu(d)$

# CS theorem for WMCFG

**Theorem (CS theorem for CFL [CS63])**

*Let $L$ be a language. T.f.a.e.*

1. *$L$ is a CFL, and*
2. *there are a regular language $R$, a Dyck language $D$ and a string homomorphism $h$ such that $L = h(R \cap D)$.*

# CS theorem for WMCFG

**Theorem (CS theorem for weighted CFL [DV14])**

*Let $L$ be a language. T.f.a.e.*

1. *$L$ is a weighted CFL, and*
2. *there are a regular language $R$, a Dyck language $D$ and a weighted string homomorphism $h$ such that $L = h(R \cap D)$.*

# CS theorem for WMCFG

**Theorem (CS theorem for weighted multiple CFL [Den16])**

*Let $L$ be a language. T.f.a.e.*

1. *$L$ is a weighted multiple CFL, and*
2. *there are a regular language $R$, a multiple Dyck language $D$ and a weighted string homomorphism $h$ such that $L = h(R \cap D)$.*

# $k$-best CS parsing algorithm for WMCFG

**Input:** $W$-weighted MCFG $(G, \mu)$, $w$, total order $\trianglelefteq$ on $W$, $k \in \mathbb{N}$
**Output:** sequence $d_1 \ldots d_k$ of ASTs for $w$ in $G$ s.t. $\mu(d_1) \trianglelefteq \ldots \trianglelefteq \mu(d_k)$ and $\nexists d\colon \mu(d_k) \ntrianglelefteq \mu(d)$

1: construct $\Delta$, $R \subseteq \Delta^*$, $D \subseteq \Delta^*$, $h\colon \Delta^* \to (\Sigma^* \to W)$ w.r.t. $[\![G, \mu]\!]$
2: $R^w \leftarrow \{v \in \Delta^* \mid h(v)(w) \neq 0\}$
3: $R^{local}\colon R \cap R^w \to W$ such that $R^{local}(v) = h(v)(w)$
4: **return** $\big(\mathrm{ordered}_{\trianglelefteq}\ ;\ \mathrm{filter}(D)\ ;\ \mathrm{map}(\mathrm{toderiv})\ ;\ \mathrm{take}(k)\big)(R^{local})$

# the regular language $R$

$$\rho_1 = S \to \langle \quad x_1^1 \quad x_2^1 \quad x_1^2 \quad x_2^2 \quad \rangle(A, B)$$

# the regular language $R$

$$\rho_1 = S \to \langle \quad [\![^1_{\rho_1} (^1_{\rho_1,1} \quad x^1_1 \quad )^1_{\rho_1,1} (^1_{\rho_1,2} \quad x^1_2 \quad )^1_{\rho_1,2} (^2_{\rho_1,1} \quad x^2_1 \quad )^2_{\rho_1,1} (^2_{\rho_1,2} \quad x^2_2 \quad )^2_{\rho_1,2} ]\!]^1_{\rho_1} \quad \rangle(A, B)$$

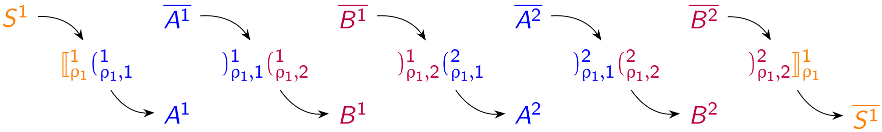
# the regular language $R$

$$S^1 \quad \overline{A^1} \quad \overline{B^1} \quad \overline{A^2} \quad \overline{B^2}$$

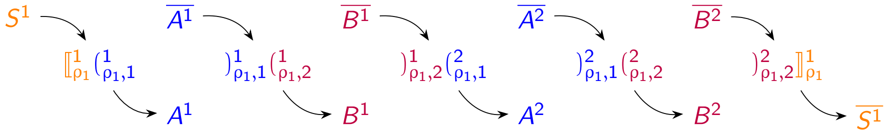
$$\rho_1 = S \to \langle \; \llbracket^1_{\rho_1} (^1_{\rho_1,1} \; x^1_1 \; )^1_{\rho_1,1} (^1_{\rho_1,2} \; x^1_2 \; )^1_{\rho_1,2} (^2_{\rho_1,1} \; x^2_1 \; )^2_{\rho_1,1} (^2_{\rho_1,2} \; x^2_2 \; )^2_{\rho_1,2} \rrbracket^1_{\rho_1} \; \rangle (A, B)$$

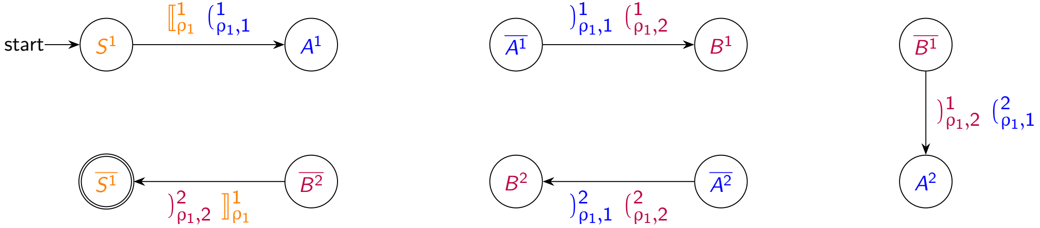
$$A^1 \quad B^1 \quad A^2 \quad B^2 \quad \overline{S^1}$$

# the regular language $R$



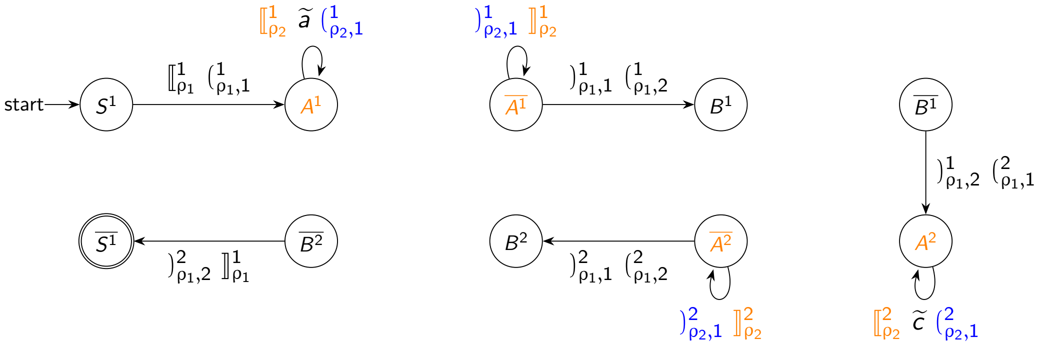
# the regular language $R$



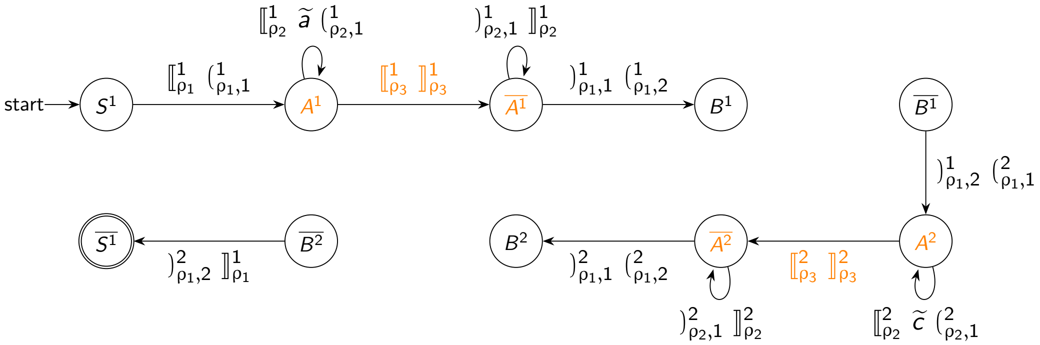
# the regular language $R$

$$\rho_2 = A \rightarrow \langle a x_1^1, c x_1^2 \rangle (A)$$



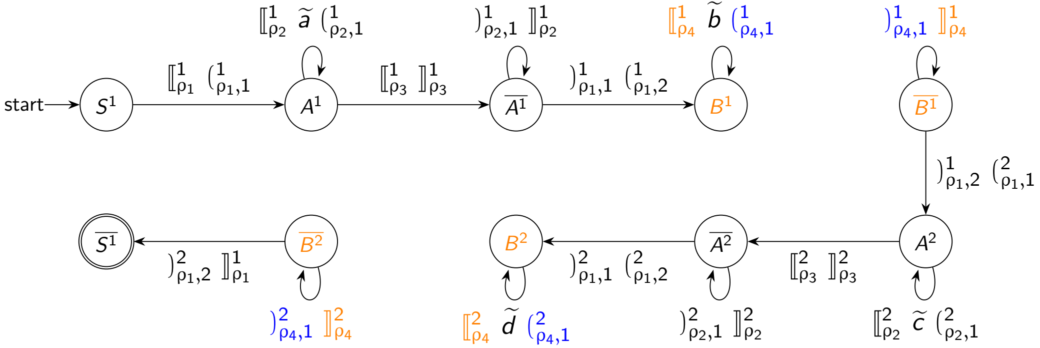
# the regular language $R$

$$\rho_3 = A \rightarrow \langle \varepsilon, \varepsilon \rangle ()$$



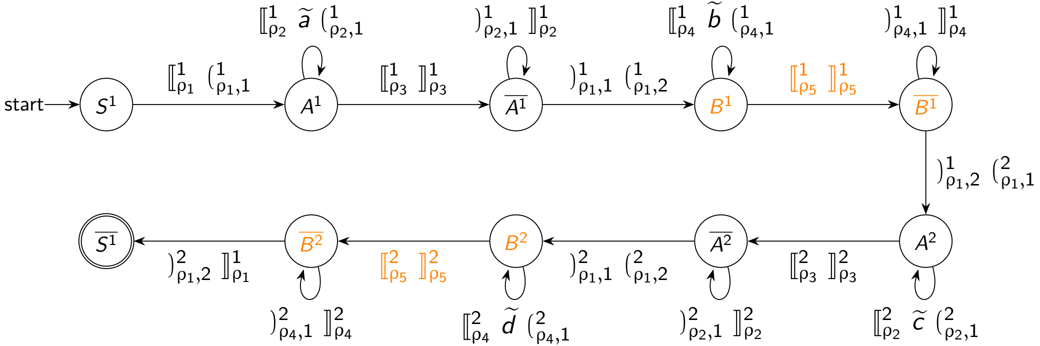
# the regular language $R$

$$\rho_4 = B \to \langle b x_1^1, d x_1^2 \rangle (B)$$



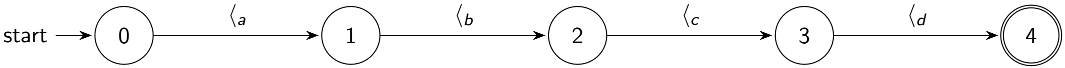
# the regular language $R$

$$\rho_5 = B \to \langle \varepsilon, \varepsilon \rangle ()$$



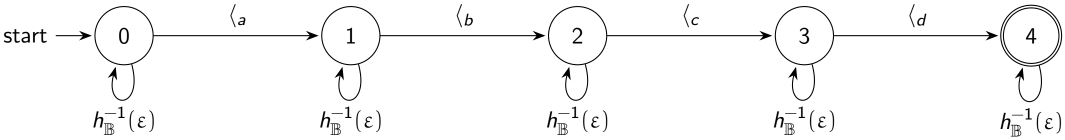
# the regular language $R^{abcd}$

- accept bracket words $v$ with $h(v)(abcd) \neq 0$



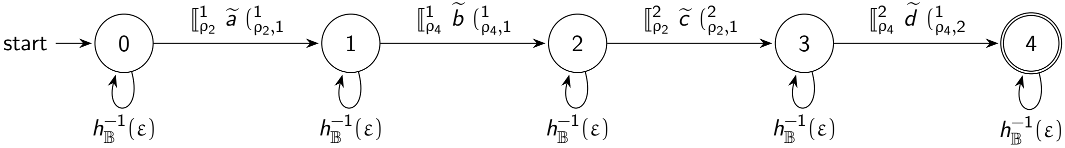
# the regular language $R^{abcd}$

- accept bracket words $v$ with $h(v)(abcd) \neq 0$
- $h_{\mathbb{B}}^{-1}(\varepsilon)$ contains brackets w/o terminal symbols



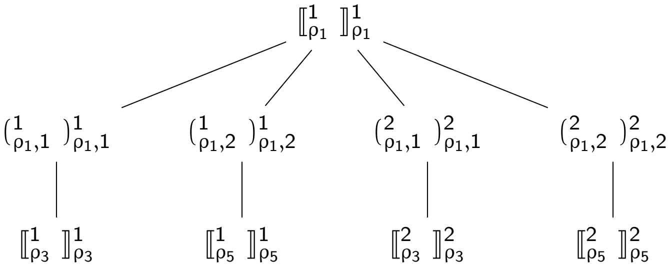
# the regular language $R^{abcd}$

- accept bracket words $v$ with $h(v)(abcd) \neq 0$
- $h_{\mathbb{B}}^{-1}(\varepsilon)$ contains brackets w/o terminal symbols
- we know context of from $R$



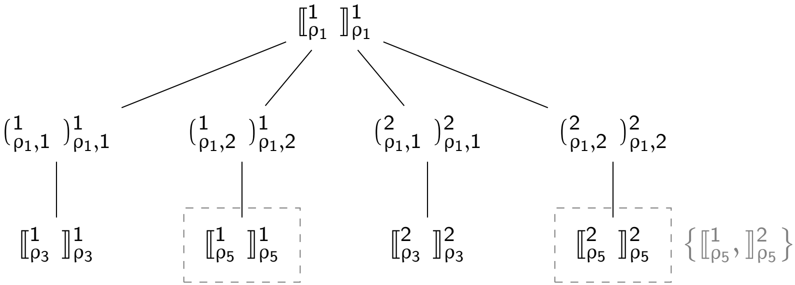
# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$



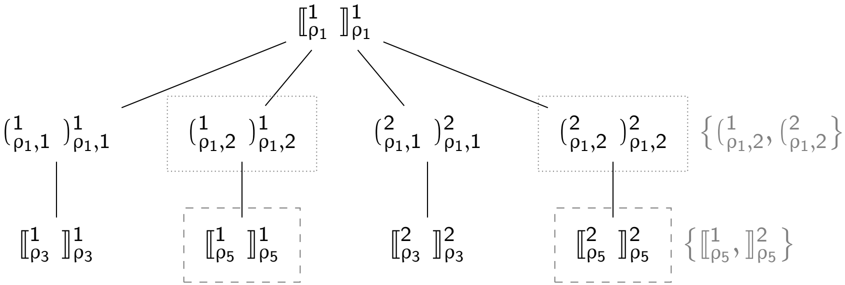
# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$
- cancel parentheses of same group simultaneously



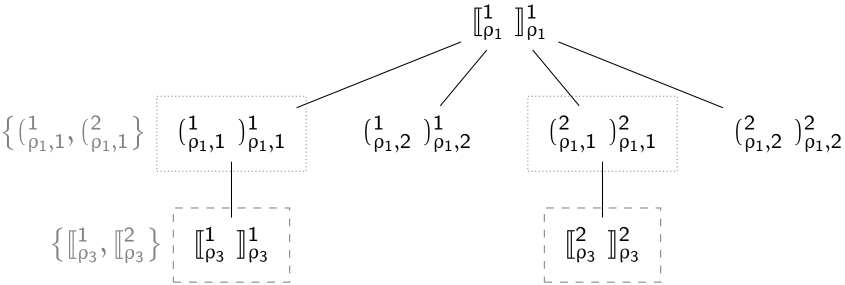
# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$
- cancel parentheses of same group simultaneously
- parent parentheses must be in one group



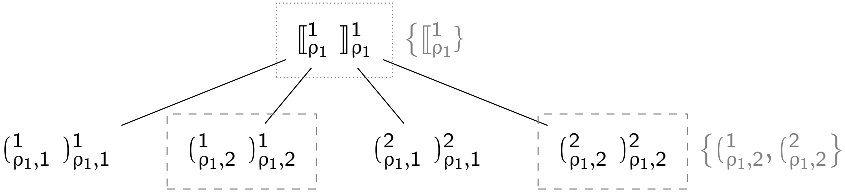
# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$
- cancel parentheses of same group simultaneously
- parent parentheses must be in one group



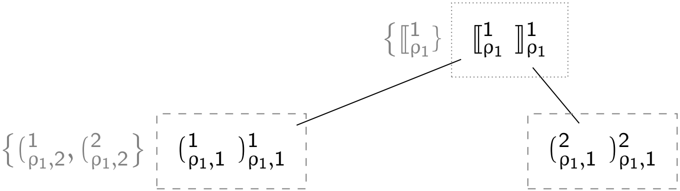
# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$
- cancel parentheses of same group simultaneously
- parent parentheses must be in one group



# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$
- cancel parentheses of same group simultaneously
- parent parentheses must be in one group



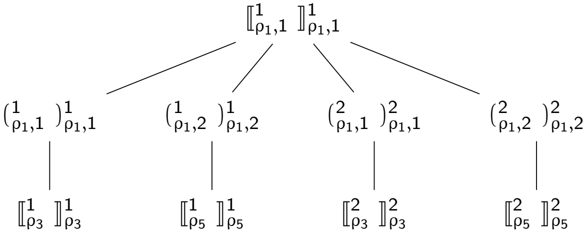
# the multiple Dyck language $D$: cancelation rule

- MDL over $\Delta$ is characterized by a partition $\mathfrak{P}$
- cancel parentheses of same group simultaneously
- parent parentheses must be in one group

$$\llbracket^1_{\rho_1} \; \rrbracket^1_{\rho_1} \quad \{\llbracket^1_{\rho_1}\}$$

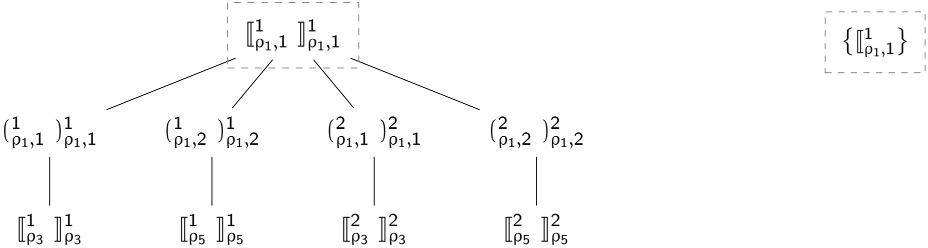
# the multiple Dyck language $D$: compact parenthesis hierarchy

- combine nodes w/ parentheses of same group



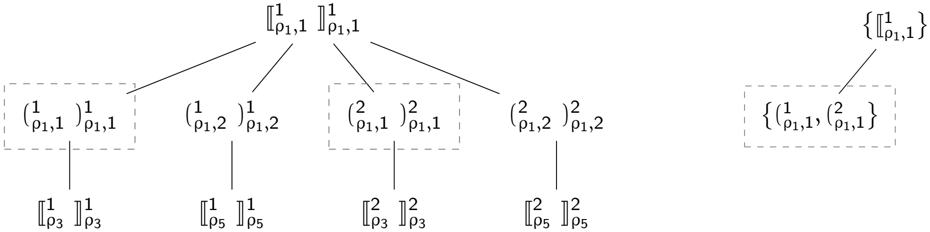
# the multiple Dyck language $D$: compact parenthesis hierarchy

- combine nodes w/ parentheses of same group



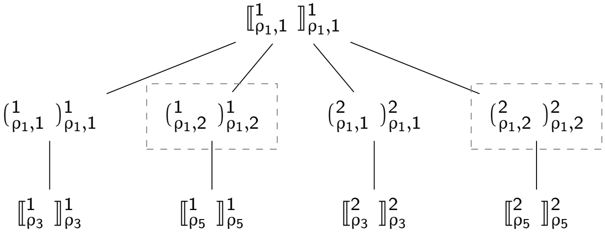
# the multiple Dyck language $D$: compact parenthesis hierarchy

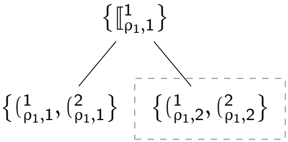
- combine nodes w/ parentheses of same group



# the multiple Dyck language $D$: compact parenthesis hierarchy

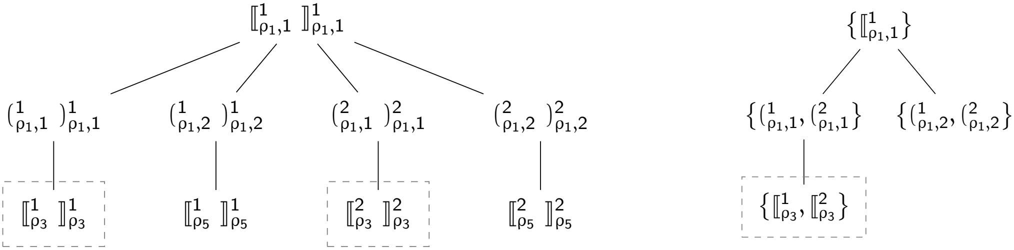
- combine nodes w/ parentheses of same group



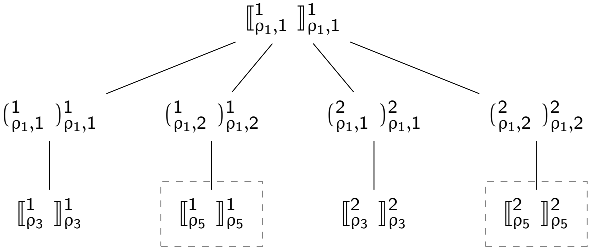
# the multiple Dyck language $D$: compact parenthesis hierarchy

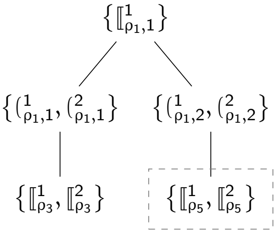
- combine nodes w/ parentheses of same group



# the multiple Dyck language $D$: compact parenthesis hierarchy

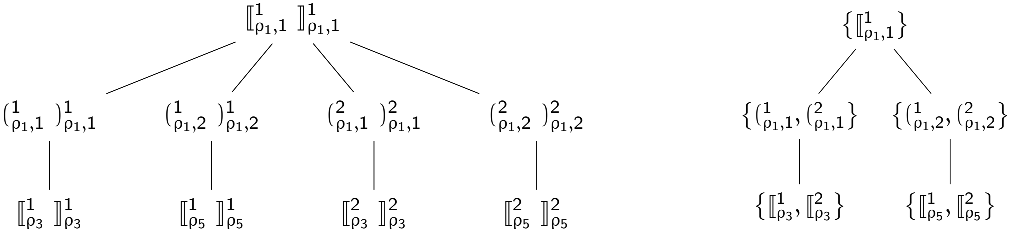
- combine nodes w/ parentheses of same group



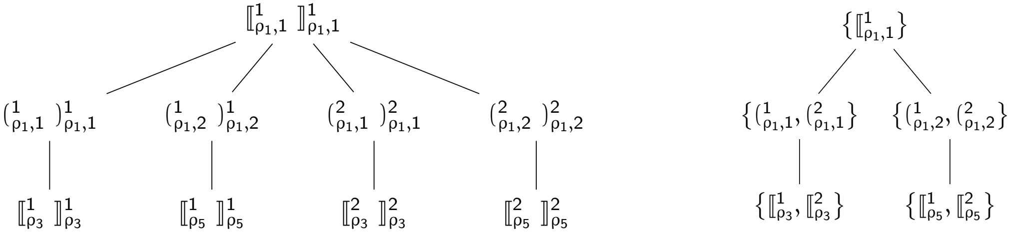
# the multiple Dyck language $D$: compact parenthesis hierarchy

- combine nodes w/ parentheses of same group
- shows relation to abstract syntax tree



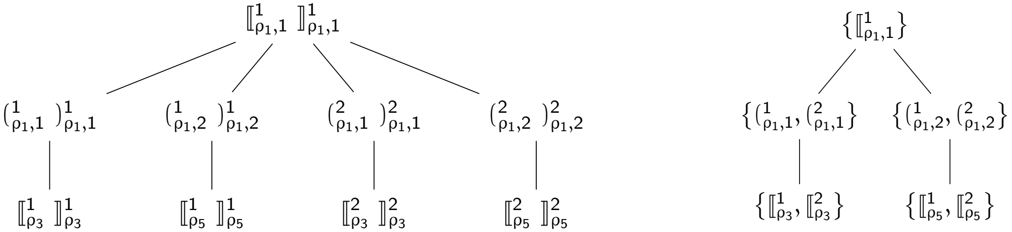
# the multiple Dyck language $D$: compact parenthesis hierarchy

- combine nodes w/ parentheses of same group
- shows relation to abstract syntax tree
- structure relates to recognition w/ TSA



# the multiple Dyck language $D$: compact parenthesis hierarchy

- combine nodes w/ parentheses of same group
- shows relation to abstract syntax tree
- structure relates to recognition w/ TSA
- observation: special structure of words in $R \cap D$



# outline

weighted multiple context-free grammars

CS parsing for WMCFG

implementation

evaluation results

conclusions and future work

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes
  - deterministic FSA and PDA

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes
  - deterministic FSA and PDA
  - product constructions of PDA with FSA and FSA with FSA

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes
  - deterministic FSA and PDA
  - product constructions of PDA with FSA and FSA with FSA
  - enumeration of language of FSA and PDA

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes
  - deterministic FSA and PDA
  - product constructions of PDA with FSA and FSA with FSA
  - enumeration of language of FSA and PDA
  - class of TSA that recognizes multiple Dyck languages

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes
  - deterministic FSA and PDA
  - product constructions of PDA with FSA and FSA with FSA
  - enumeration of language of FSA and PDA
  - class of TSA that recognizes multiple Dyck languages
  - automata constructions for the CS parser (for $R$, $R^w$ and $D$)

[1] `github.com/tud-fop/rustomata`
[2] `github.com/truprecht/openfsa`

# implementation

- implementation in *Rust*, as a part of *Rustomata*[1]
- task includes implementation of Rust interface for *OpenFst*[2], abandoned due to
  - lack of generative functionality,
  - optimizations introduce PDA
- implementation includes
  - deterministic FSA and PDA
  - product constructions of PDA with FSA and FSA with FSA
  - enumeration of language of FSA and PDA
  - class of TSA that recognizes multiple Dyck languages
  - automata constructions for the CS parser (for $R$, $R^w$ and $D$)
  - conversion from MCFG to non-deleting MCFG

[1] github.com/tud-fop/rustomata
[2] github.com/truprecht/openfsa

# outline

weighted multiple context-free grammars

CS parsing for WMCFG

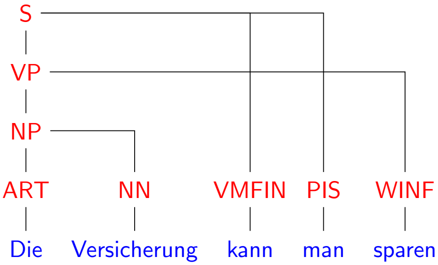
implementation

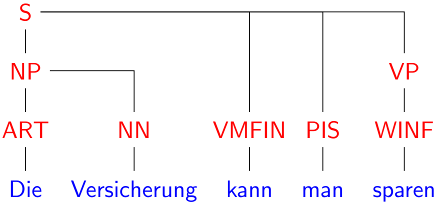
evaluation results

conclusions and future work

# qualitative evaluation metric: parseval [Bla+91; Col97]

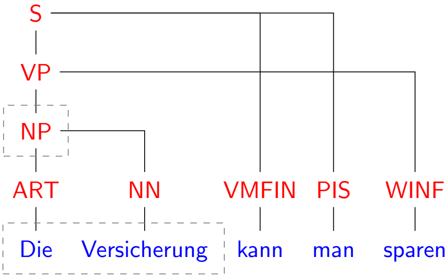
- match *constituents by span*, report precision, recall and f score

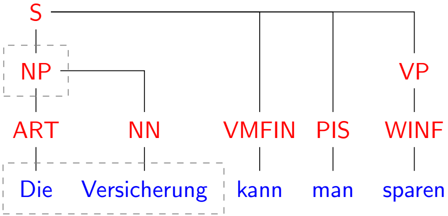


# qualitative evaluation metric: parseval [Bla+91; Col97]

- match *constituents by span*, report precision, recall and f score

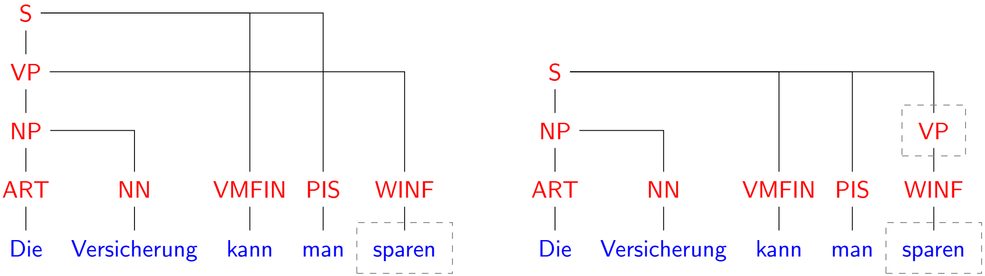



*TP*: I *FP*: *FN*:

# qualitative evaluation metric: parseval [Bla+91; Col97]

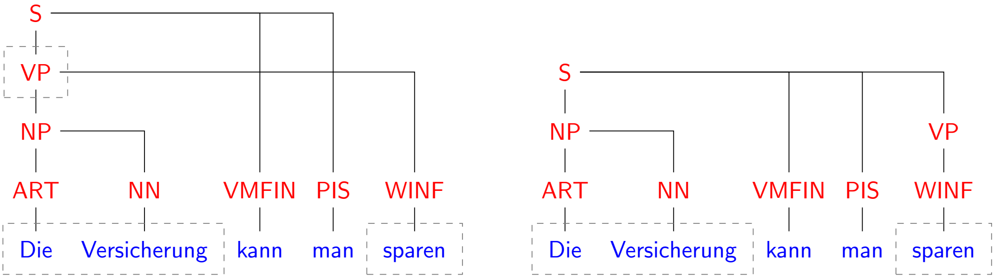
- match *constituents by span*, report precision, recall and f score




*TP*: I *FP*: I *FN*:

# qualitative evaluation metric: parseval [Bla+91; Col97]

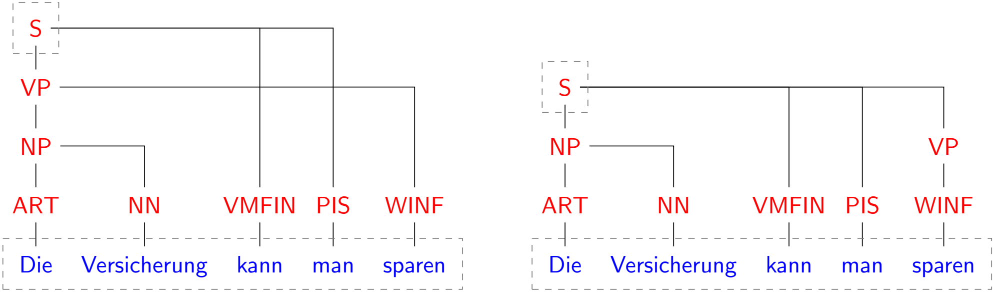
- match *constituents by span*, report precision, recall and f score




*TP*: I *FP*: I *FN*: I

# qualitative evaluation metric: parseval [Bla+91; Col97]

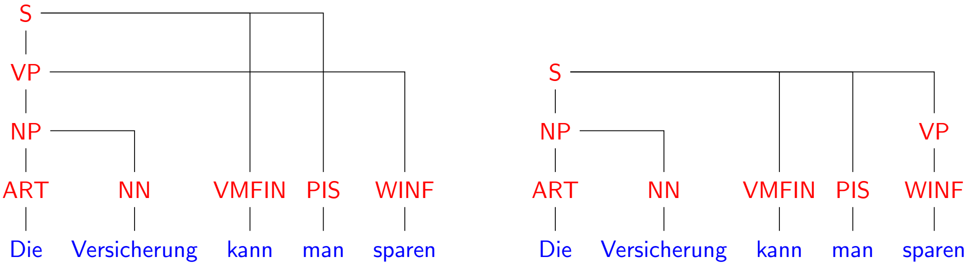
- match *constituents by span*, report precision, recall and f score




*TP*: I I *FP*: I *FN*: I

# qualitative evaluation metric: parseval [Bla+91; Col97]

- match *constituents by span*, report precision, recall and f score

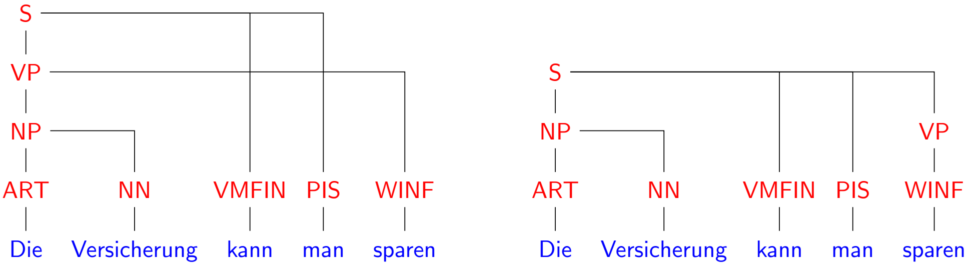



*TP*: I I *FP*: I *FN*: I

$$precision = \frac{TP}{TP+FP} = \frac{2}{3} \qquad recall = \frac{TP}{TP+FN} = \frac{2}{3}$$

# qualitative evaluation metric: parseval [Bla+91; Col97]

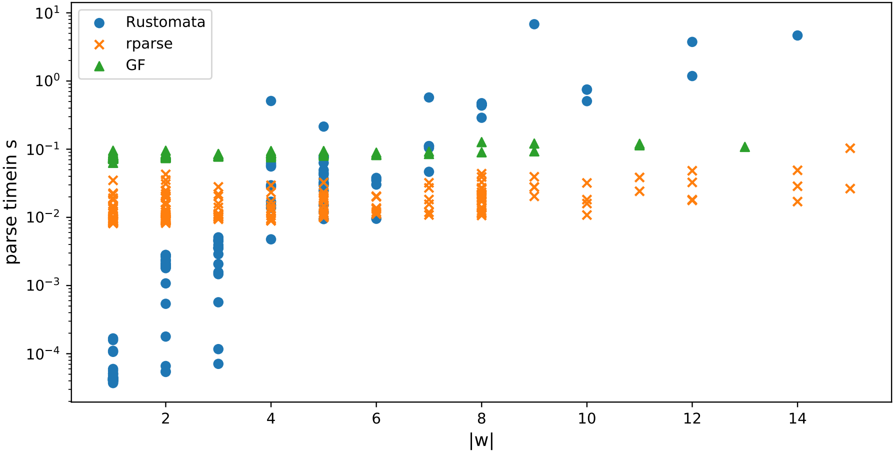
- match *constituents by span*, report precision, recall and f score
- accuracy of predicted *pos labels*



# results: quality

| parser | Rustomata | rparse | Grammatical Framework |
|---|---|---|---|
| accuracy of POS-tags | 0.64 | 0.94 | 0.44 |
| number of predicted constituents | 306 | 416 | 239 |
| labeled precision for all constituents | 0.96 | 0.86 | 0.94 |
| labeled recall for all constituents | 0.70 | 0.86 | 0.62 |
| labeled f-score for all constituents | 0.81 | 0.86 | 0.75 |

# results: parse time



# outline

weighted multiple context-free grammars

CS parsing for WMCFG

implementation

evaluation results

conclusions and future work

# conclusions and future work

- implementation works with natural language corpora

# conclusions and future work

- implementation works with natural language corpora
- resolved some performance issues by more restrictive definitions
  - context-free alternative to $R$
  - filter language with minimal set of parenthesis
  - sorted multiple Dyck languages

# conclusions and future work

- implementation works with natural language corpora
- resolved some performance issues by more restrictive definitions
  - context-free alternative to $R$
  - filter language with minimal set of parenthesis
  - sorted multiple Dyck languages
- implementation was a lot of work
  - originally planned to use OpenFst, dropped later
  - implementation of (push-down and finite state) automata with heuristic search

# conclusions and future work

- implementation works with natural language corpora
- resolved some performance issues by more restrictive definitions
  - context-free alternative to $R$
  - filter language with minimal set of parenthesis
  - sorted multiple Dyck languages
- implementation was a lot of work
  - originally planned to use OpenFst, dropped later
  - implementation of (push-down and finite state) automata with heuristic search
- compared to rparse and Grammatical Framework, our parser
  - has comparable accuracy, but
  - inferior parse time

# conclusions and future work

- implementation works with natural language corpora
- resolved some performance issues by more restrictive definitions
  - context-free alternative to $R$
  - filter language with minimal set of parenthesis
  - sorted multiple Dyck languages
- implementation was a lot of work
  - originally planned to use OpenFst, dropped later
  - implementation of (push-down and finite state) automata with heuristic search
- compared to rparse and Grammatical Framework, our parser
  - has comparable accuracy, but
  - inferior parse time
- possible future work
  - further investigate search items in the generative approach
  - find efficient automata frameworks for FSA and PDA or improve implementation
  - $R \cap D$ by product construction of automata

# references


[Bla+91] Ezra Black, Steven Abney, Dan Flickenger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, et al. "A procedure for quantitatively comparing the syntactic coverage of English grammars". In: *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*. 1991.

[Col97] Michael Collins. "Three generative, lexicalised models for statistical parsing". In: *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 1997, pp. 16–23.

[CS63] Noam Chomsky and Marcel P Schützenberger. "The algebraic theory of context-free languages". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 35. Elsevier, 1963, pp. 118–161.

[Den16] Tobias Denkinger. "A Chomsky-Schützenberger representation for weighted multiple context-free languages". In: *arXiv preprint arXiv:1606.03982* (2016).

[DV14] Manfred Droste and Heiko Vogler. "The Chomsky-Schützenberger theorem for quantitative context-free languages". In: *International Journal of Foundations of Computer Science* 25.08 (2014), pp. 955–969.

[Hul09] Mans Hulden. "Parsing CFGs and PCFGs with a Chomsky-Schützenberger representation". In: *Language and Technology Conference*. Springer. 2009, pp. 151–160.

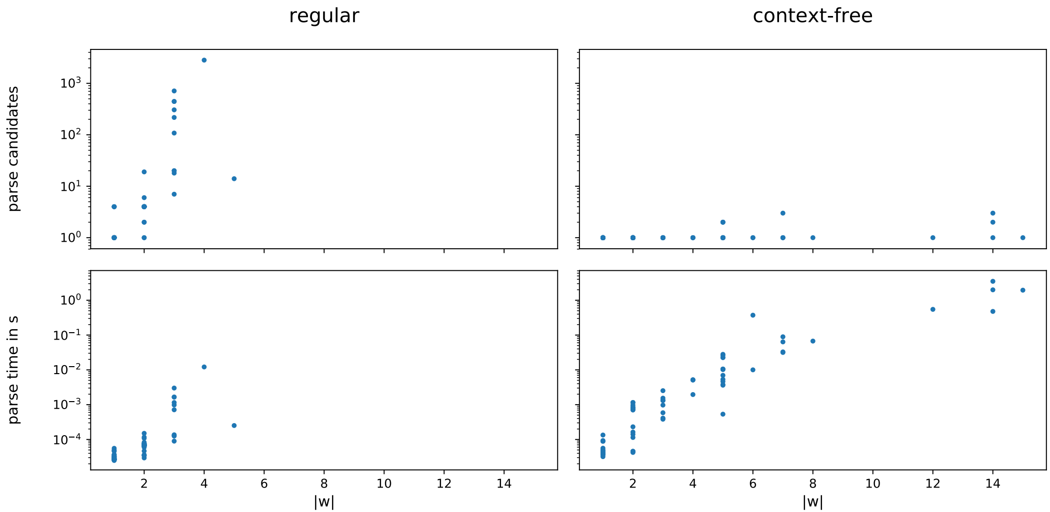
[Sek+91] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. "On multiple context-free grammars". In: *Theoretical Computer Science* 88.2 (1991), pp. 191–229.

# BACKUP

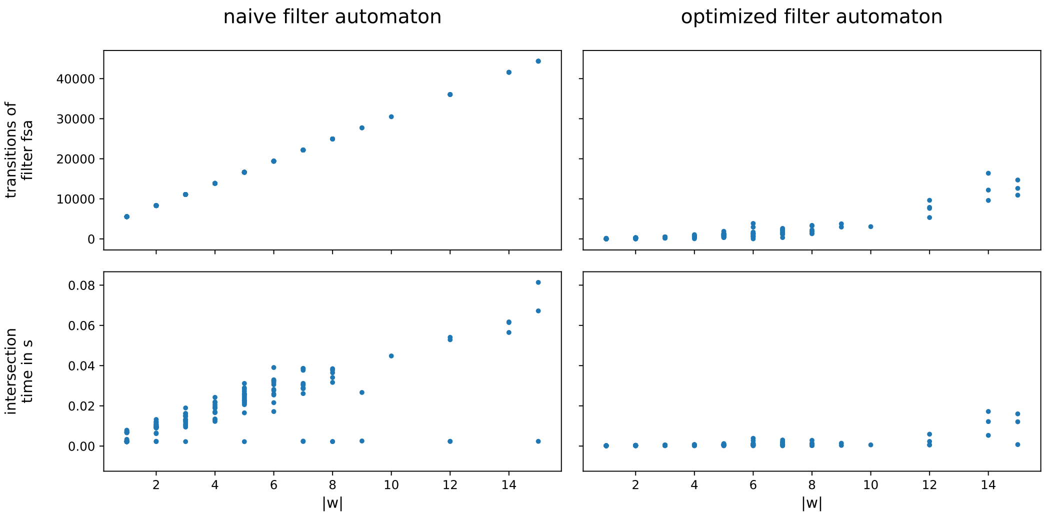
# factorizable bimonoids

- Viterbi: $([0,1], \max, \cdot, 0, 1)$, where $\sqrt[n]{w} \cdot \ldots \cdot \sqrt[n]{w}$ is a $(\geq, n)$-factorization of $w \in [0,1]$
- tropic: $(R^{\infty}_{\geq 0}, \min, +, \infty, 0)$, where $\frac{w}{n} + \ldots + \frac{w}{n}$ is a $(\leq, n)$-factorization of $w \in R^{\infty}_{\geq 0}$
- boolean: $(\{\mathrm{true}, \mathrm{false}\}, \vee, \wedge, \mathrm{false}, \mathrm{true})$, where $\mathrm{true} \cdot \ldots \cdot \mathrm{true}$ is a $(\trianglelefteq, n)$-factorization of $\mathrm{true}$ and $\mathrm{false} \cdot \ldots \cdot \mathrm{false}$ is a $(\trianglelefteq, n)$-factorization of $\mathrm{false}$

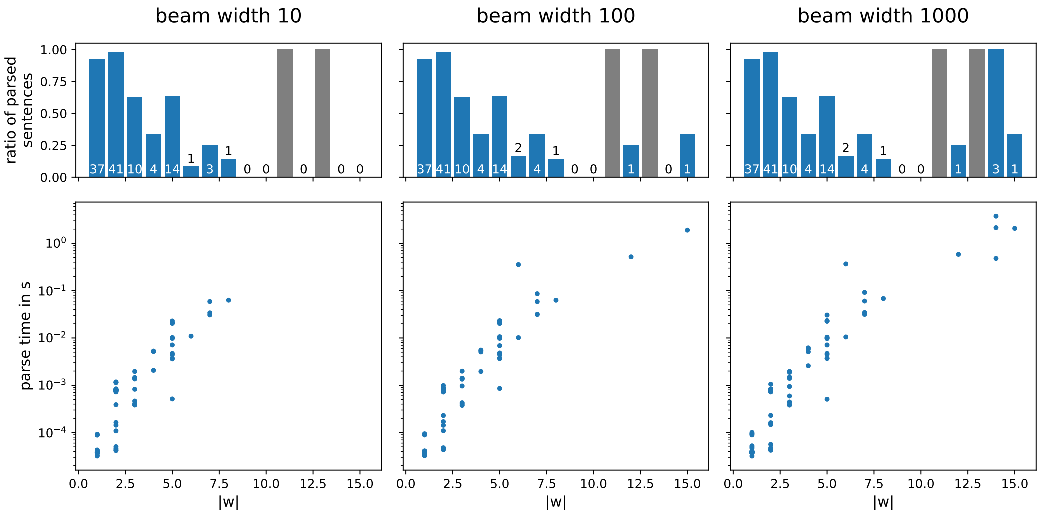
# results: optimized generator language $R$



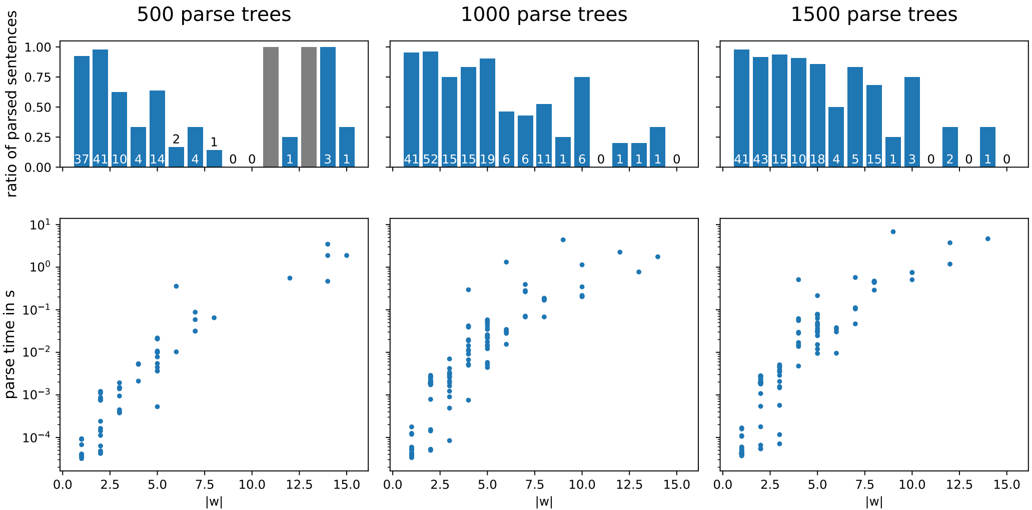
# results: optimized filter language $R^w$



# results: beam search

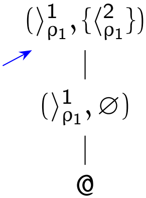


# results: grammar size



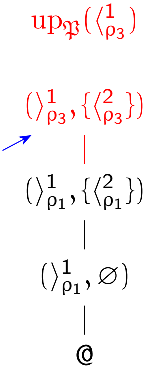
# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$
  - $\mathrm{down}(\delta)$



# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$
  - $\mathrm{down}(\delta)$



# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$
  - $\mathrm{down}(\delta)$

$\mathrm{down}(\rangle^{1}_{\rho_3})$

$(-, \{\langle^{2}_{\rho_3}\})$

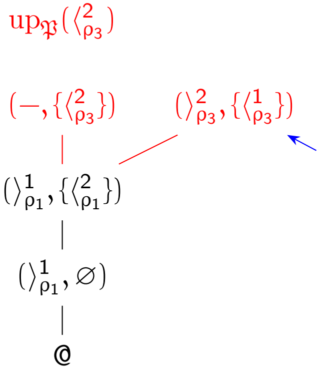
$(\rangle^{1}_{\rho_1}, \{\langle^{2}_{\rho_1}\})$

$(\rangle^{1}_{\rho_1}, \varnothing)$

@

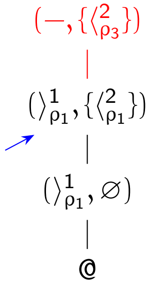
# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$
  - $\mathrm{down}(\delta)$



# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$ (nondeterministic)
  - $\mathrm{down}(\delta)$



# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$ (nondeterministic)
  - $\mathrm{down}(\delta)$

or $\mathrm{up}_{\mathfrak{P}}(\langle^{2}_{\rho_3})$

$$(\rangle^{2}_{\rho_3}, \varnothing)$$

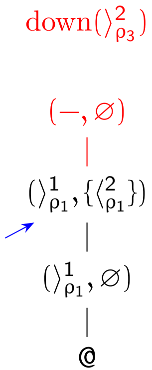
$$(\rangle^{1}_{\rho_1}, \{\langle^{2}_{\rho_1}\})$$

$$(\rangle^{1}_{\rho_1}, \varnothing)$$

@

# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$ (nondeterministic)
  - $\mathrm{down}(\delta)$

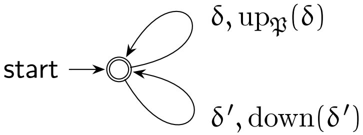


# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$ (nondeterministic)
  - $\mathrm{down}(\delta)$
- accepting configuration only contains root symbol ($@$) and symbols o.t.f. $(-, \varnothing)$

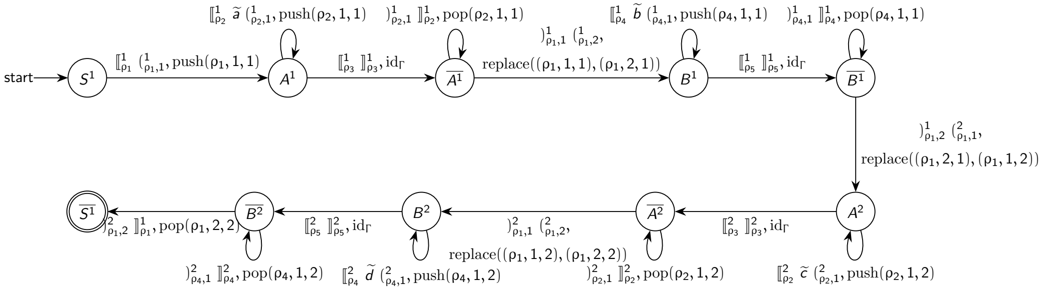
# recognizing $D$ using a TSA

- tree stack instructions
  - $\mathrm{up}_{\mathfrak{P}}(\delta)$ (nondeterministic)
  - $\mathrm{down}(\delta)$
- accepting configuration only contains root symbol (@) and symbols o.t.f. $(-, \varnothing)$
- for each $\delta \in \Delta, \delta' \in \overline{\Delta}$



# $R \cap \mathrm{D}(\Delta)$

- there are *lots* of candidates, very few are even Dyck words
- limit $R$ to Dyck words using PDA
- superset approximation of generator PDA is generator FSA



# omit unnecessary brackets

- omit all grammar rules that cannot be utilized in derivation

$$
\begin{aligned}
P:\ & S \rightarrow [x_{1,1}x_{2,1}x_{1,2}x_{2,2}](A,B),\\
& A \rightarrow [x_{1,1}a, x_{1,2}c](A),\\
& A \rightarrow [\varepsilon,\varepsilon](),\\
& B \rightarrow [x_{1,1}b, x_{1,2}d](B),\\
& B \rightarrow [\varepsilon,\varepsilon]()
\end{aligned}
$$

$$
\begin{aligned}
P_\varepsilon:\ & S \rightarrow [x_{1,1}x_{2,1}x_{1,2}x_{2,2}](A,B)\\
& A \rightarrow [\varepsilon,\varepsilon]()\\
& B \rightarrow [\varepsilon,\varepsilon]()
\end{aligned}
$$

# mutliple Dyck language

- congruence relation for each $v_1, \ldots, v_k \in D(\Sigma)$ with $v_1 \ldots v_k \equiv_{\Sigma,\mathfrak{P}} \varepsilon$

  $$\sigma_1 v_1 \overline{\sigma_1} u_1 \sigma_2 \ldots u_{k-1} \sigma_k v_k \overline{\sigma_k} \equiv_{\Sigma,\mathfrak{P}} u_1 \ldots u_{k-1}$$

  if $\{\sigma_1, \ldots, \sigma_k\} \in \mathfrak{P}$ and $u_1, \ldots, u_k$ are Dyck words over $\Sigma$
- $D = [\varepsilon]_{\equiv_{\Delta,\mathfrak{P}(G)}}$

# sorted multiple Dyck language

- $(\Sigma, \mathit{sort})$ sorted alphabet, $\mathfrak{P}$ partition of $\Sigma$ such that $\mathit{sort}(\sigma) = \mathit{sort}(\sigma') \implies \exists \mathfrak{p} \in \mathfrak{P}\colon \sigma, \sigma' \in \mathfrak{p}$
- let $\sigma_1, \ldots, \sigma_k \in \Sigma$, $v_1, \ldots, v_k \in \mathrm{D}(\Sigma)$, $\sigma_1 v_1 \overline{\sigma_1} \ldots \sigma_k v_k \overline{\sigma_k} \in \mathrm{smD}(\Sigma, \mathit{sort}, \mathfrak{P})$ iff there is a partition $\mathfrak{B}$ of $[k]$ such that for each $\{b_1, \ldots, b_\ell\} \in \mathfrak{B}$ where $b_1 < \ldots < b_\ell$:
  - $v_{b_1} \ldots v_{b_\ell} \in \mathrm{smD}(\Sigma, \mathit{sort}, \mathfrak{P})$
  - $\{\sigma_{b_1}, \ldots, \sigma_{b_\ell}\} \in \mathfrak{P}$
  - for each $\{b'_1, \ldots, b'_{\ell'}\} \in \mathfrak{B}$: $\mathit{sort}(\{\sigma_{b_1}, \ldots, \sigma_{b_\ell}\}) = \mathit{sort}(\{b'_1, \ldots, b'_{\ell'}\}) \iff \{b'_1, \ldots, b'_{\ell'}\} = \{\sigma_{b_1}, \ldots, \sigma_{b_\ell}\}$

# reachability analysis

- let $G = (N, \Sigma, P, S)$ non-deleting MCFG and $w \in \Sigma^*$
- set of $w$-productive rules $P_w$ is the smallest set $P' \subseteq P$ such that

  $$\rho = A \rightarrow [u_{1,0}x_{i(1,1)}^{j(1,1)}u_{1,1}\dots x_{i(1,m_1)}^{j(1,m_1)}u_{1,m_1},\dots,u_{s,0}x_{i(s,1)}^{j(s,1)}\dots u_{s,m_s}](A_1,\dots,A_k) \in P'$$

  iff $\rho \in P$, $u_{1,0},\dots,u_{s,m_s}$ are subsequences in $w$ and there are rules with lhs $A_1,\dots,A_k \in P'$