Extracting semi-Dyck words from fsa using the CYK algorithm

Thomas Ruprecht

November 30, 2018

Outline

Motivation: Chomsky-Schützenberger parsing

- ChoSchü theorem [CS63]: decompose context-free language into
  - reg. language R
  - alph. string homomorphism h
  - semi-Dyck language D

  such that $L = h(R \cap D)$

Motivation: Chomsky-Schützenberger parsing

- ChoSchü theorem [CS63]: decompose context-free language into
  - reg. language R
  - alph. string homomorphism h
  - semi-Dyck language D

  such that $L = h(R \cap D)$
- ChoSchü parsing [Hul11]:
  - def. of R and D using grammar imply
    - bijection between $R \cap D$ and derivation trees
    - bijection between $R \cap D \cap h^{-1}(w)$ and derivation trees for w

Motivation: Chomsky-Schützenberger parsing

- ChoSchü theorem [CS63]: decompose context-free language into
  - reg. language R
  - alph. string homomorphism h
  - semi-Dyck language D

  such that $L = h(R \cap D)$
- ChoSchü parsing [Hul11]:
  - def. of R and D using grammar imply
    - bijection between $R \cap D$ and derivation trees
    - bijection between $R \cap D \cap h^{-1}(w)$ and derivation trees for w
  - goal: extract semi-Dyck words from reg. language $R \cap h^{-1}(w)$

Motivation: existing algorithm to extract Dyck words [Hul11]

Require: finite state automaton $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\mathsf{init}}, q_{\mathsf{fin}}, T)$
Ensure: enumerate words in $\mathrm{L}(\mathcal{A}) \cap \mathrm{D}(\Sigma)$

procedure $\textsc{extractDyck}(\mathcal{A})$
  $A, C := \{v \mid (p, \sigma, q), (q, \overline{\sigma}, r) \in T\}, \emptyset$
  for $(p, v, q) \in A$ **do**
    $A \setminus= \{(p, v, q)\}; C \cup= \{(p, v, q)\}$
    if $(p, q) = (q_{\mathsf{init}}, q_{\mathsf{fin}})$ **then yield** v
    $A \cup= \{(p, vw, r) \mid (q, w, r) \in C\} \setminus C$
    $A \cup= \{(o, uv, q) \mid (o, u, p) \in C\} \setminus C$
    $A \cup= \{(o, \sigma v \overline{\sigma}, r) \mid (o, \sigma, p), (q, \overline{\sigma}, r) \in T\} \setminus C$

Motivation: existing algorithm to extract Dyck words [Hul11]

Require: finite state automaton $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\mathsf{init}}, q_{\mathsf{fin}}, T)$

Ensure: enumerate words in $\mathrm{L}(\mathcal{A}) \cap \mathrm{D}(\Sigma)$

procedure $\textsc{extractDyck}(\mathcal{A})$

$\quad A, C := \{v \mid (p, \sigma, q), (q, \overline{\sigma}, r) \in T\}, \emptyset$

$\quad$ **for** $(p, v, q) \in A$ **do**

$\quad\quad A \setminus= \{(p, v, q)\}; C \cup= \{(p, v, q)\}$

$\quad\quad$ **if** $(p, q) = (q_{\mathsf{init}}, q_{\mathsf{fin}})$ **then yield** v

$\quad\quad A \cup= \{(p, vw, r) \mid (q, w, r) \in C\} \setminus C$

$\quad\quad A \cup= \{(o, uv, q) \mid (o, u, p) \in C\} \setminus C$

$\quad\quad A \cup= \{(o, \sigma v \overline{\sigma}, r) \mid (o, \sigma, p), (q, \overline{\sigma}, r) \in T\} \setminus C$

- relies on recursive structure of Dyck words: concatenation and bracketing

Motivation: existing algorithm to extract Dyck words [Hul11]

Require: finite state automaton $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\text{init}}, q_{\text{fin}}, T)$

Ensure: enumerate words in $\mathrm{L}(\mathcal{A}) \cap \mathrm{D}(\Sigma)$

procedure $\textsc{extractDyck}(\mathcal{A})$

$\quad A, C := \{v \mid (p, \sigma, q), (q, \overline{\sigma}, r) \in T\}, \emptyset$

$\quad$ **for** $(p, v, q) \in A$ **do**

$\quad\quad A \setminus= \{(p, v, q)\}; C \cup= \{(p, v, q)\}$

$\quad\quad$ **if** $(p, q) = (q_{\text{init}}, q_{\text{fin}})$ **then yield** v

$\quad\quad A \cup= \{(p, vw, r) \mid (q, w, r) \in C\} \setminus C$

$\quad\quad A \cup= \{(o, uv, q) \mid (o, u, p) \in C\} \setminus C$

$\quad\quad A \cup= \{(o, \sigma v \overline{\sigma}, r) \mid (o, \sigma, p), (q, \overline{\sigma}, r) \in T\} \setminus C$

- relies on recursive structure of Dyck words: concatenation and bracketing
- dynamic programming: store intermediate results (backlinks) for state

Motivation: existing algorithm to extract Dyck words [Hul11]

Require: finite state automaton $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\mathsf{init}}, q_{\mathsf{fin}}, T)$

Ensure: enumerate words in $\mathrm{L}(\mathcal{A}) \cap \mathrm{D}(\Sigma)$

procedure $\textsc{extractDyck}(\mathcal{A})$

  $A, C := \{v \mid (p, \sigma, q), (q, \overline{\sigma}, r) \in T\}, \emptyset$

  for $(p, v, q) \in A$ **do**

    $A \setminus= \{(p, v, q)\}; C \cup= \{(p, v, q)\}$

    if $(p, q) = (q_{\mathsf{init}}, q_{\mathsf{fin}})$ **then yield** v

    $A \cup= \{(p, vw, r) \mid (q, w, r) \in C\} \setminus C$

    $A \cup= \{(o, uv, q) \mid (o, u, p) \in C\} \setminus C$

    $A \cup= \{(o, \sigma v \overline{\sigma}, r) \mid (o, \sigma, p), (q, \overline{\sigma}, r) \in T\} \setminus C$

- relies on recursive structure of Dyck words: concatenation and bracketing
- dynamic programming: store intermediate results (backlinks) for state
- backlinks are equivalent to reduct grammar [BPS61]

Outline

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,][\![\{\}]\!]\Big)\Big\}$ is 3-centered

n-centered semi-Dyck languages

- example $[\,(\,)\,]\left\{\left(\,[\,]\,[\![\,\{\,\}\,]\!]\,\right)\right\}$ is 3-centered

n-centered semi-Dyck languages

- example $[\,(\,)\,]\left\{\left(\,[\,]\,[\![\,\{\,\}\,]\!]\,\right)\right\}$ is 3-centered

n-centered semi-Dyck languages

- example $[\,(\,)\,]\left\{\left(\,[\,]\,[\![\,\{\,\}\,]\!]\,\right)\right\}$ is 3-centered

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,][\![\{\}]\!]\Big)\Big\}$ is 3-centered

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,]⟦\{\}⟧\Big)\Big\}$ is 3-centered

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,]⟦\{\}⟧\Big)\Big\}$ is 3-centered
- n-centered semi-Dyck word o.t.f. $w_0 (_1)_1 w_1 ... (_n)_n w_n$ where
  - $w_i \in \overline{\Sigma}^* \cdot \Sigma^*$
  - $w_0 (_1)_1 w_1 ... (_n)_n w_n \in \mathrm{D}(\Sigma)$

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,][\![\{\}]\!]\Big)\Big\}$ is 3-centered
- n-centered semi-Dyck word o.t.f. $w_0(_1)_1w_1 \ldots (_n)_nw_n$ where
  - $w_i \in \overline{\Sigma}^* \cdot \Sigma^*$
  - $w_0(_1)_1w_1 \ldots (_n)_nw_n \in \mathrm{D}(\Sigma)$
- $\mathrm{C}(\Sigma, n) \subseteq \mathrm{D}(\Sigma)$

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,]⟦\{\}⟧\Big)\Big\}$ is 3-centered
- n-centered semi-Dyck word o.t.f. $w_0 (_1)_1 w_1 \dots (_n)_n w_n$ where
  - $w_i \in \overline{\Sigma}^* \cdot \Sigma^*$
  - $w_0 (_1)_1 w_1 \dots (_n)_n w_n \in \mathrm{D}(\Sigma)$
- $\mathrm{C}(\Sigma, n) \subseteq \mathrm{D}(\Sigma)$
- $\mathrm{C}(\Sigma, \leq n) = \bigcup_{n' \leq n} \mathrm{C}(\Sigma, n')$

n-centered semi-Dyck languages

- example $[()]\Big\{\Big([\,][\![\{\}]\!]\Big)\Big\}$ is 3-centered
- n-centered semi-Dyck word o.t.f. $w_0(_1)_1w_1\ldots(_n)_nw_n$ where
  - $w_i \in \overline{\Sigma}^* \cdot \Sigma^*$
  - $w_0(_1)_1w_1\ldots(_n)_nw_n \in \mathrm{D}(\Sigma)$
- $\mathrm{C}(\Sigma, n) \subseteq \mathrm{D}(\Sigma)$
- $\mathrm{C}(\Sigma, \leq n) = \bigcup_{n' \leq n} \mathrm{C}(\Sigma, n')$
- $\mathrm{C}(\Sigma, \leq \infty) = \bigcup_{n' \in \mathbb{N}} \mathrm{C}(\Sigma, n') = \mathrm{D}(\Sigma)$

(At most) n-centered regular languages

- n -centered regular word o.t.f.
  $w_0 (_1)_1 w_1 ... (_n)_n w_n$
  w_i does not contain subsequences in $\Sigma \cdot \overline{\Sigma}$

(At most) n-centered regular languages

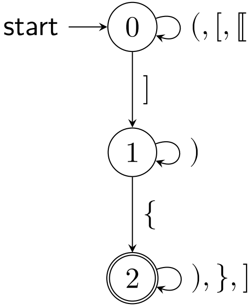
- n -centered regular word o.t.f.
  $w_0 (_1)_1 w_1 \ldots (_n)_n w_n$
  w_i does not contain subsequences in $\Sigma \cdot \overline{\Sigma}$
- $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\text{init}}, q_{\text{fin}}, T)$ is n -centered
  - surjective function $f\colon Q \to \{0, \ldots, n\}$:
    $$(p, \sigma, q) \in T \Rightarrow \begin{cases} f(p) = f(r) - 1 & \text{if } (q, \overline{\sigma}, r) \in T \\ f(p) = f(q) & \text{otherwise} \end{cases}$$
    vice versa for $(p, \overline{\sigma}, q)$
  - $=$ state partition with ordered cells

(At most) n-centered regular languages

- n -centered regular word o.t.f.
  $w_0 (_1)_1 w_1 \dots (_n)_n w_n$
  w_i does not contain subsequences in $\Sigma \cdot \overline{\Sigma}$
- $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\text{init}}, q_{\text{fin}}, T)$ is n -centered
  - surjective function $f \colon Q \to \{0, \dots, n\}$:

    $$(p, \sigma, q) \in T \Rightarrow \begin{cases} f(p) = f(r) - 1 & \text{if } (q, \overline{\sigma}, r) \in T \\ f(p) = f(q) & \text{otherwise} \end{cases}$$

    vice versa for $(p, \overline{\sigma}, q)$
  - $=$ state partition with ordered cells



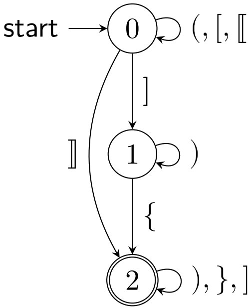
(At most) n-centered regular languages

- $(\leq n)$-centered regular word o.t.f.
  $w_0 (_1)_1 w_1 \ldots (_m)_m w_m$ where $m \leq n$,
  w_i does not contain subsequences in $\Sigma \cdot \overline{\Sigma}$
- $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\text{init}}, q_{\text{fin}}, T)$ is $(\leq n)$-centered

  function $f\colon Q \to \{0, \ldots, n\}$:

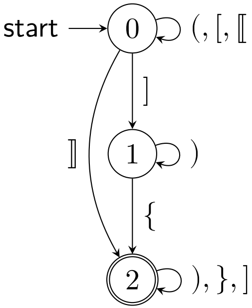
  $$(p, \sigma, q) \in T \Rightarrow \begin{cases} f(p) < f(r) & \text{if } (q, \overline{\sigma}, r) \in T \\ f(p) = f(q) & \text{otherwise} \end{cases}$$

  vice versa for $(p, \overline{\sigma}, q)$
  - $\approx$ state partition with ordered cells



(At most) n-centered regular languages

- $(\leq n)$-centered regular word o.t.f.
  $w_0 (_1)_1 w_1 \ldots (_m)_m w_m$ where $m \leq n$,
  w_i does not contain subsequences in $\Sigma \cdot \overline{\Sigma}$
- $\mathcal{A} = (Q, \Sigma \cup \overline{\Sigma}, q_{\text{init}}, q_{\text{fin}}, T)$ is $(\leq n)$-centered
  function $f \colon Q \to \{0, \ldots, n\}$:
  $$(p, \sigma, q) \in T \Rightarrow \begin{cases} f(p) < f(r) & \text{if } (q, \overline{\sigma}, r) \in T \\ f(p) = f(q) & \text{otherwise} \end{cases}$$
  vice versa for $(p, \overline{\sigma}, q)$
  - $\approx$ state partition with ordered cells
  - $\hat{n}$ smallest number s.t. $\mathcal{A}$ is $(\leq \hat{n})$-centered $\Rightarrow$ f is surjective



Closure properties

L is $(\leq \ell)$-centered, M is $(\leq m)$-centered reg. language over Σ, for $\ell, m \in \mathbb{N} \cup \{\infty\}$

- $L \cap M$
- $L \cup M$
- $\overline{L}$
- $L \setminus M$
- $L \cap \mathrm{D}(\Sigma)$

Closure properties

L is $(\leq \ell)$-centered, M is $(\leq m)$-centered reg. language over Σ, for $\ell, m \in \mathbb{N} \cup \{\infty\}$

- $L \cap M$ is $(\leq \min(\ell, m))$-centered
- $L \cup M$
- $\overline{L}$
- $L \setminus M$
- $L \cap \mathrm{D}(\Sigma)$

Closure properties

L is $(\leq \ell)$-centered, M is $(\leq m)$-centered reg. language over Σ, for $\ell, m \in \mathbb{N} \cup \{\infty\}$

- $L \cap M$ is $(\leq \min(\ell, m))$-centered
- $L \cup M$ is $(\leq \max(\ell, m))$-centered
- $\overline{L}$
- $L \setminus M$
- $L \cap \mathrm{D}(\Sigma)$

Closure properties

L is $(\leq \ell)$-centered, M is $(\leq m)$-centered reg. language over Σ, for $\ell, m \in \mathbb{N} \cup \{\infty\}$

- $L \cap M$ is $(\leq \min(\ell, m))$-centered
- $L \cup M$ is $(\leq \max(\ell, m))$-centered
- $\overline{L}$ is $(\leq \infty)$-centered
- $L \setminus M$
- $L \cap \mathrm{D}(\Sigma)$

Closure properties

L is $(\leq \ell)$-centered, M is $(\leq m)$-centered reg. language over Σ, for $\ell, m \in \mathbb{N} \cup \{\infty\}$

- $L \cap M$ is $(\leq \min(\ell, m))$-centered
- $L \cup M$ is $(\leq \max(\ell, m))$-centered
- $\overline{L}$ is $(\leq \infty)$-centered
- $L \setminus M$ is $(\leq \ell)$-centered
- $L \cap \mathrm{D}(\Sigma)$

Closure properties

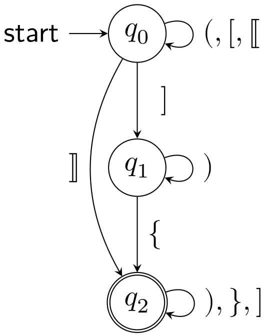
L is $(\leq \ell)$-centered, M is $(\leq m)$-centered reg. language over Σ, for $\ell, m \in \mathbb{N} \cup \{\infty\}$

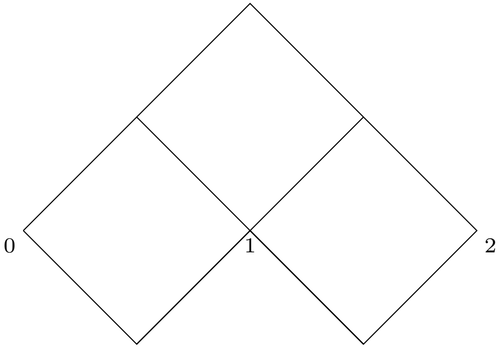
- $L \cap M$ is $(\leq \min(\ell, m))$-centered
- $L \cup M$ is $(\leq \max(\ell, m))$-centered
- $\overline{L}$ is $(\leq \infty)$-centered
- $L \setminus M$ is $(\leq \ell)$-centered
- $L \cap \mathrm{D}(\Sigma) \subseteq \mathrm{C}(\Sigma, \leq \ell)$

Outline

CYK algorithm for extraction of semi-Dyck words: example

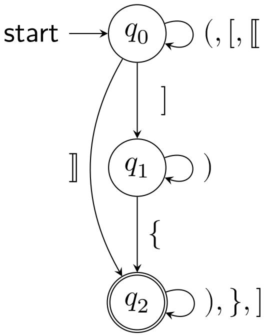
- n-CYK algorithm applicable for $(\leq n)$-centered automata

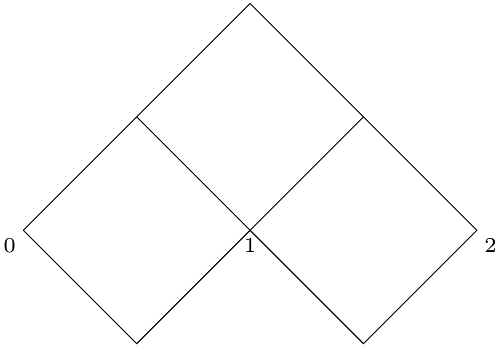




CYK algorithm for extraction of semi-Dyck words: example

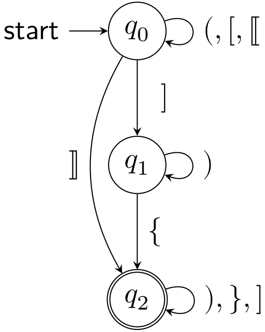
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words

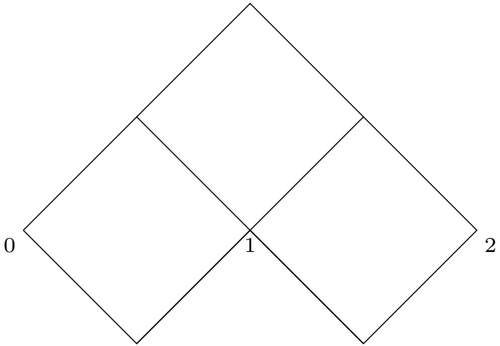




CYK algorithm for extraction of semi-Dyck words: example

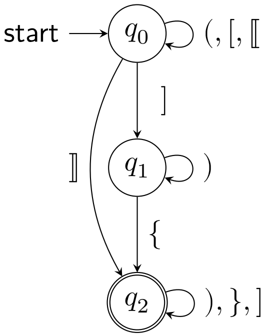
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$

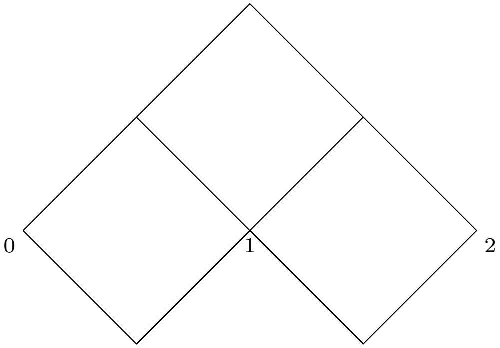




CYK algorithm for extraction of semi-Dyck words: example

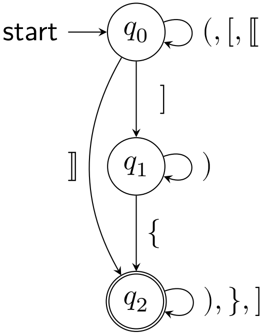
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$
  - concatenation: $o, r \to (o, p)(p, r)$

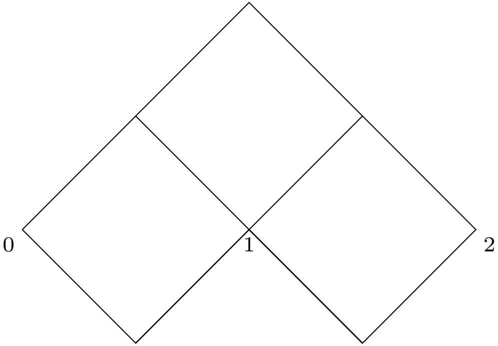




CYK algorithm for extraction of semi-Dyck words: example

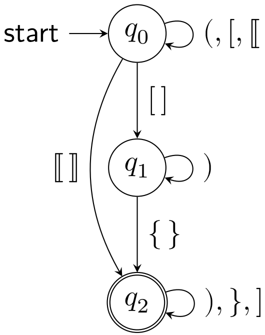
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$
  - concatenation: $o, r \to (o, p)(p, r)$
  - bracketing: $o, r \to \sigma(p, q)\overline{\sigma}$ for $(o, \sigma, p), (q, \overline{\sigma}, r) \in T$

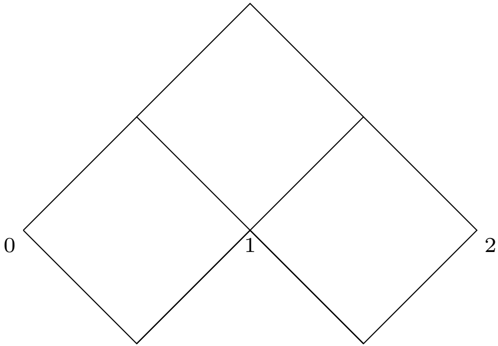




CYK algorithm for extraction of semi-Dyck words: example

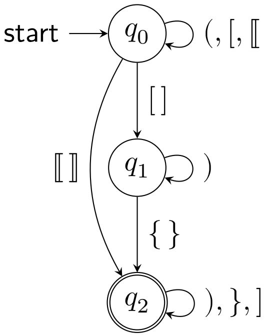
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$
  - concatenation: $o, r \to (o, p)(p, r)$
  - bracketing: $o, r \to \sigma(p, q)\overline{\sigma}$ for $(o, \sigma, p), (q, \overline{\sigma}, r) \in T$

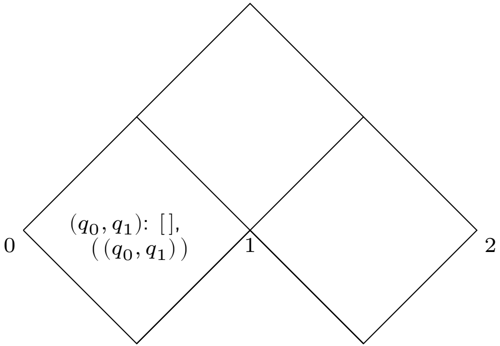




CYK algorithm for extraction of semi-Dyck words: example

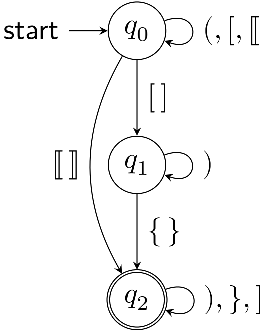
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$
  - concatenation: $o, r \to (o, p)(p, r)$
  - bracketing: $o, r \to \sigma(p, q)\overline{\sigma}$ for $(o, \sigma, p), (q, \overline{\sigma}, r) \in T$

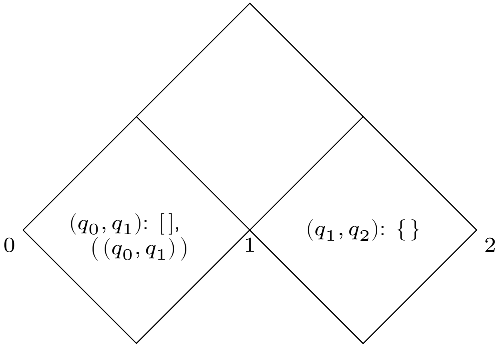




CYK algorithm for extraction of semi-Dyck words: example

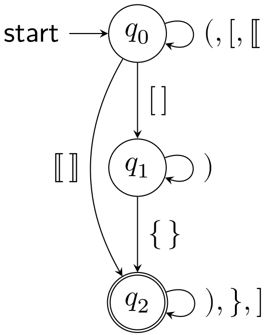
- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$
  - concatenation: $o, r \to (o, p)(p, r)$
  - bracketing: $o, r \to \sigma(p, q)\overline{\sigma}$ for $(o, \sigma, p), (q, \overline{\sigma}, r) \in T$

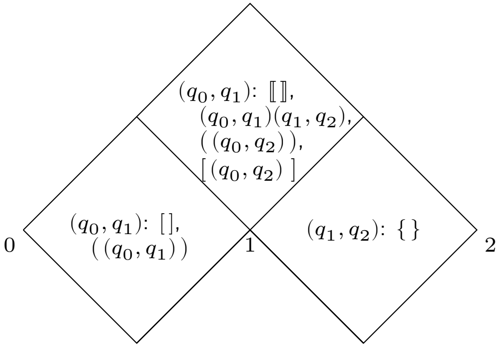




CYK algorithm for extraction of semi-Dyck words: example

- n-CYK algorithm applicable for $(\leq n)$-centered automata
- span $f(o), f(r)$: fill backlinks for sub-runs accepting semi-Dyck words
  - initial: $o, r \to \sigma\overline{\sigma}$ for $(o, \sigma, p), (p, \overline{\sigma}, r) \in T$
  - concatenation: $o, r \to (o, p)(p, r)$
  - bracketing: $o, r \to \sigma(p, q)\overline{\sigma}$ for $(o, \sigma, p), (q, \overline{\sigma}, r) \in T$





CYK algorithm for extraction of semi-Dyck words

Require: $n_{\geq}$-centered automaton $\mathcal{A} = (Q, \Sigma, q_{\text{init}}, q_{\text{fin}}, T)$
Ensure: enumerates elements of $\mathcal{L}(\mathcal{A}) \cap D(\Sigma)$

procedure EXTRACTDYCK($\mathcal{A}$)
    $\mathcal{A}' :=$ NORMALFORM($\mathcal{A}$) ▷ combine transitions $(o, \sigma, p), (p, \overline{\sigma}, q)$ to $(o, \sigma\overline{\sigma}, q)$
    $C :=$ CYK($\mathcal{A}'$)
    ENUMERATE($C, q_{\text{init}}, q_{\text{fin}}$) ▷ c.f. Huang and Chiang [HC05]
function CYK($\mathcal{A}$)
    for $r \in \{1, \ldots, n\}$ **do**
        for $l \in \{0, \ldots, n-1\}$ **do**
            $S_{l,l+r} := \{(p, q) \mid (p, \sigma\overline{\sigma}, q) \in T, f(p) = l, f(q) = l + r\}$
            for $m \in \{1, \ldots, r-1\}$ **do**
                $S_{l,l+r} \cup= \{(o, q) \mid (o, p) \in S_{l,m}, (p, q) \in S_{m,l+r}\}$
            $S_{l,l+r} \cup= \bigcup_{(p,q) \in S_{l,l+r}} \mathrm{R}_{\mathcal{A}}(p, q)$ ▷ transitively reachable (o, r) via $(o, \sigma, p), (q, \overline{\sigma}, r) \in T$
    return $(S_{i,j} \mid i \in \{0, \ldots, n-1\}, j \in \{i+1, \ldots, n\})$

Conclusion

- application for Chomsky-Schützenberger parsing [Hul11; Den17]:
  - R, $h^{-1}(w)$ are $(\leq\infty)$-centered

Conclusion

- application for Chomsky-Schützenberger parsing [Hul11; Den17]:
  - R, $h^{-1}(w)$ are $(\leq\infty)$-centered
  - $R \cap h^{-1}(w)$ is $(\leq|w|)$-centered for ε-free grammars

Conclusion

- application for Chomsky-Schützenberger parsing [Hul11; Den17]:
  - R, $h^{-1}(w)$ are $(\leq\infty)$-centered
  - $R \cap h^{-1}(w)$ is $(\leq|w|)$-centered for ε-free grammars
  - size of closure $\mathrm{R}_{\mathcal{A}}(p,q)$ depends on chain rules

Conclusion

- application for Chomsky-Schützenberger parsing [Hul11; Den17]:
  - R, $h^{-1}(w)$ are $(\leq\infty)$-centered
  - $R \cap h^{-1}(w)$ is $(\leq|w|)$-centered for ε-free grammars
  - size of closure $\mathrm{R}_{\mathcal{A}}(p,q)$ depends on chain rules
- CYK parsing of cfg without binarization

Conclusion

- application for Chomsky-Schützenberger parsing [Hul11; Den17]:
  - R, $h^{-1}(w)$ are $(\leq\infty)$-centered
  - $R \cap h^{-1}(w)$ is $(\leq|w|)$-centered for ε-free grammars
  - size of closure $\mathrm{R}_{\mathcal{A}}(p, q)$ depends on chain rules
- CYK parsing of cfg without binarization
- closure properties:
  - parse multiple words at same time
  - even using different grammars

References

[BPS61] Yehoshua Bar-Hillel, Micha Asher Perles, and Eli Shamir. "On Formal Properties of Simple Phrase Structure Grammars". In: *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14 (1–4 1961), pp. 143–172. ISSN: 1867-8319. DOI: 10.1524/stuf.1961.14.14.143.

[CS63] Noam Chomsky and Marcel Paul Schützenberger. "The algebraic theory of context-free languages". In: *Computer Programming and Formal Systems, Studies in Logic* (1963), pp. 118–161. DOI: 10.1016/S0049-237X(09)70104-1.

[Den17] Tobias Denkinger. "Chomsky-Schützenberger parsing for weighted multiple context-free languages". In: *Journal of Language Modelling* 5.1 (July 2017), p. 3. DOI: 10.15398/jlm.v5i1.159.

[HC05] Liang Huang and David Chiang. "Better k-best Parsing". In: *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005, pp. 53–64. URL: http://dl.acm.org/citation.cfm?id=1654494.1654500.

[Hul11] Mans Hulden. "Parsing CFGs and PCFGs with a Chomsky-Schützenberger Representation". In: *Human Language Technology. Challenges for Computer Science and Linguistics*. Ed. by Zygmunt Vetulani. Vol. 6562. Lecture Notes in Computer Science. 2011, pp. 151–160. ISBN: 978-3-642-20094-6. DOI: 10.1007/978-3-642-20095-3_14.