Chomsky-Schützenberger parsing for weighted multiple context-free languages

Tobias Denkinger

Freitagsseminar 2017-10-13

Overview

1. n-best parsing
2. Multiple context-free languages (MCFLs)
3. A Chomsky-Schützenberger (CS) representation of weighted MCFLs
4. Using the CS representation for n-best parsing

n-best parsing

parsing

Input: a grammar G, a word w

Output: a derivation tree of w in G

n-best parsing

parsing

Input: a grammar G, a word w

Output: a derivation tree of w in G

n-best parsing

Input: a weighted grammar G, a word w, a number n

Output: a sequence of $\underbrace{n \text{ best}}_{\text{w.r.t. their weights}}$ derivation trees of w in G

Multiple context-free languages

a context-free grammar production

$A \to a A b$ composes strings

Multiple context-free languages

a context-free grammar production

$A \to a A b$ composes strings

a multiple context-free grammar production

$A \to [a x_1^1 b, c x_1^2 d](A)$ composes tuples of strings

Multiple context-free languages

a context-free grammar production

$A \to a A b$ composes strings

a multiple context-free grammar production

$A \to [a x_1^1 b, c x_1^2 d](A)$ composes tuples of strings

↪ extra expressive power useful for natural language processing

A CS representation for weighted MCFLs

Theorem. Let L be a weighted* language. T.f.a.e.

* over a complete commutative strong bimonoid

A CS representation for weighted MCFLs

Theorem. Let L be a weighted* language. T.f.a.e.

1. L is a weighted* MCFL.

* over a complete commutative strong bimonoid

A CS representation for weighted MCFLs

Theorem. Let L be a weighted* language. T.f.a.e.

1. L is a weighted* MCFL.
2. $\exists$ regular language R, $\exists$ multiple Dyck language D, $\exists$ weighted* α-homomorphism h s.t.

$$L = h(R \cap D).$$

* over a complete commutative strong bimonoid

A CS representation for weighted MCFLs

Theorem. Let L be a weighted* language. T.f.a.e.

1. L is a weighted* MCFL.
2. $\exists$ regular language R, $\exists$ multiple Dyck language D, $\exists$ weighted* α-homomorphism h s.t.
$$L = h(R \cap D).$$

Observation.

(1. $\Rightarrow$ 2.) decomposes L into "simpler" parts

* over a complete commutative strong bimonoid

The regular language R

$\alpha = S \to [x_1^1 \quad x_1^2](A)$

$\beta = A \to [a x_1^1 b, \; c x_1^2 d](A)$

$\gamma = A \to [\;\varepsilon\;, \;\varepsilon\;]()$

The regular language R

(idea: rule traversals)

$$\alpha = S \to [x_1^1 \quad x_1^2](A)$$

$$S \qquad \bar{S}$$

$$\beta = A \to [a\, x_1^1\, b,\ c\, x_1^2\, d](A)$$

$$A_1 \quad \bar{A}_1 \quad A_2 \quad \bar{A}_2$$

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

The regular language R

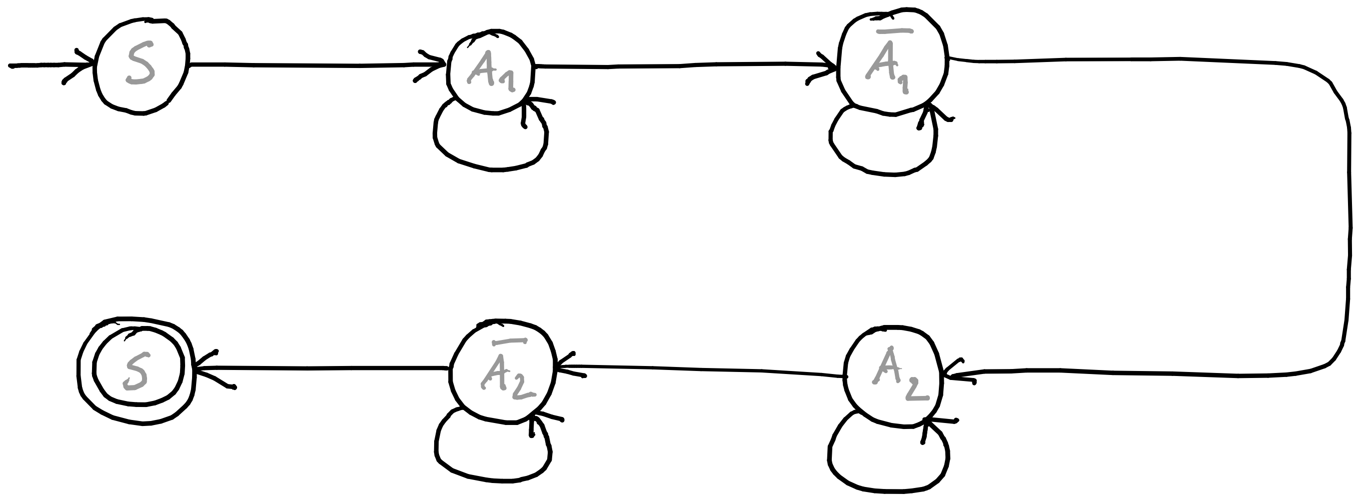
(idea: rule traversals)

$$\alpha = S \to [\underset{\uparrow S}{\,} x_1^1 \quad x_1^2 \underset{\uparrow \bar{S}}{\,}](A)$$

$$\beta = A \to [\underset{\uparrow A_1}{\,} a\, x_1^1\, b \underset{\uparrow \bar{A}_1}{\,}, \underset{\uparrow A_2}{\,} c\, x_1^2\, d \underset{\uparrow \bar{A}_2}{\,}](A)$$

$$\gamma = A \to [\overset{\downarrow A_1}{\,} \; \varepsilon \; \overset{\downarrow \bar{A}_1}{\,}, \overset{\downarrow A_2}{\,} \; \varepsilon \; \overset{\downarrow \bar{A}_2}{\,}]()$$

R:



The regular language R

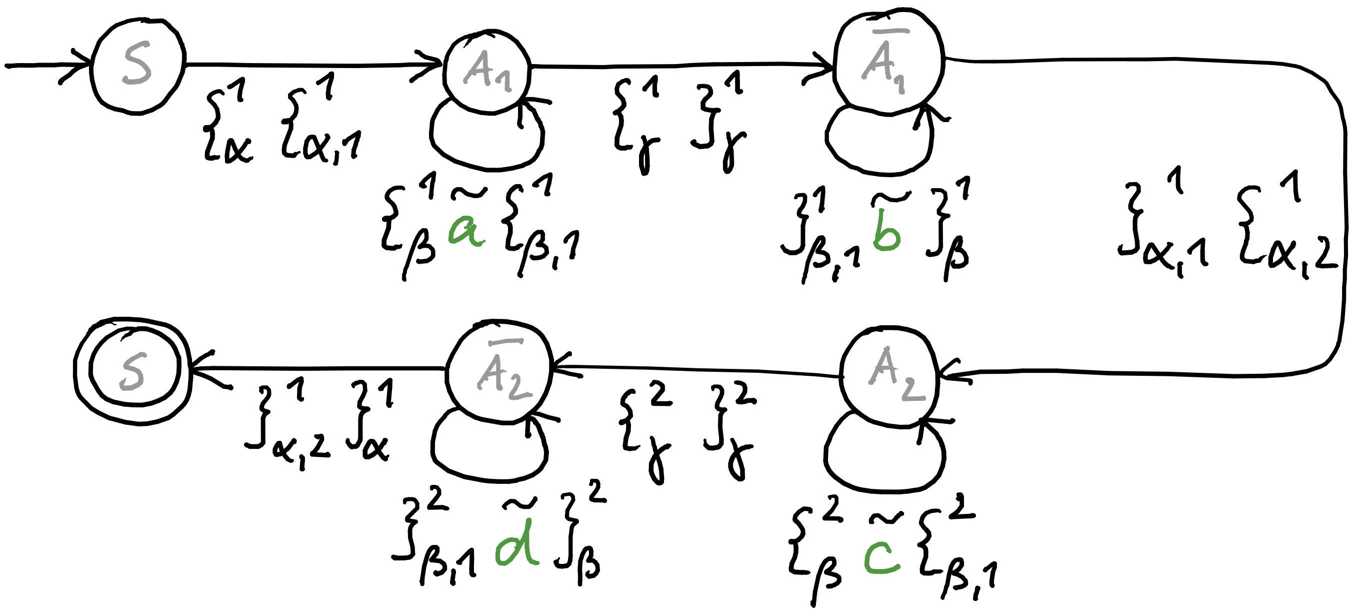
(idea: rule traversals)

$$\alpha = S \to [\underset{S}{\uparrow} x_1^1 \quad x_1^2 \underset{\bar{S}}{\uparrow}](A)$$

$$\beta = A \to [\underset{A_1}{\uparrow} a\, x_1^1\, b \underset{\bar{A}_1}{\uparrow},\ \underset{A_2}{\uparrow} c\, x_1^2\, d \underset{\bar{A}_2}{\uparrow}](A)$$

$$\gamma = A \to [\overset{A_1}{\downarrow}\ \varepsilon\ \overset{\bar{A}_1}{\downarrow},\ \overset{A_2}{\downarrow}\ \varepsilon\ \overset{\bar{A}_2}{\downarrow}]()$$

R:



The multiple Dyck language D

given by an equivalence relation $\equiv_p$

The multiple Dyck language $\mathcal{D}$

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \ldots, \sigma_k\} \in P$, u_i, v_i well-bracketed:

$$u_0 \sigma_1 v_1 \bar{\sigma}_1 u_1 \cdots \sigma_k v_k \bar{\sigma}_k u_k \equiv_P u_0 \cdots u_k \quad \text{if} \quad v_1 \cdots v_k \equiv_P \varepsilon$$

The multiple Dyck language D

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \ldots, \sigma_k\} \in P$, u_i, v_i well-bracketed:

$$u_0 \sigma_1 v_1 \bar{\sigma}_1 u_1 \cdots \sigma_k v_k \bar{\sigma}_k u_k \equiv_P u_0 \cdots u_k \quad \text{if } v_1 \cdots v_k \equiv_P \varepsilon$$

Example:

$$\{^1_\beta \quad \{^1_{\beta,1} \; \{^1_\gamma \; \}^1_\gamma \; \}^1_{\beta,1} \quad \}^1_\beta \; \{^2_\beta \quad \{^2_{\beta,1} \; \{^2_\gamma \; \}^2_\gamma \; \}^2_{\beta,1} \quad \}^2_\beta$$

The multiple Dyck language D

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \dots, \sigma_k\} \in P,\ u_i, v_i$ well-bracketed:

$$u_0 \sigma_1 v_1 \bar{\sigma}_1 u_1 \cdots \sigma_k v_k \bar{\sigma}_k u_k \equiv_P u_0 \cdots u_k \quad \text{if} \quad v_1 \cdots v_k \equiv_P \varepsilon$$

Example:

$$\{^1_\beta \quad \{^1_{\beta,1} \quad \{^1_\gamma \quad \}^1_\gamma \quad \}^1_{\beta,1} \quad \}^1_\beta \quad \{^2_\beta \quad \{^2_{\beta,1} \quad \{^2_\gamma \quad \}^2_\gamma \quad \}^2_{\beta,1} \quad \}^2_\beta$$

$$\{^1_{\beta,1} \quad \{^1_\gamma \quad \}^1_\gamma \quad \}^1_{\beta,1} \qquad \{^2_{\beta,1} \quad \{^2_\gamma \quad \}^2_\gamma \quad \}^2_{\beta,1}$$

The multiple Dyck language D

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \dots, \sigma_k\} \in P$, u_i, v_i well-bracketed:

$$u_0 \sigma_1 v_1 \overline{\sigma_1} u_1 \cdots \sigma_k v_k \overline{\sigma_k} u_k \equiv_P u_0 \cdots u_k \quad \text{if} \quad v_1 \cdots v_k \equiv_P \varepsilon$$

<u>Example:</u>

$$\{^1_\beta \quad \{^1_{\beta,1} \; \{^1_\gamma \; \}^1_\gamma \; \}^1_{\beta,1} \quad \}^1_\beta \quad \{^2_\beta \quad \{^2_{\beta,1} \; \{^2_\gamma \; \}^2_\gamma \; \}^2_{\beta,1} \quad \}^2_\beta$$

$$\{^1_{\beta,1} \; \{^1_\gamma \; \}^1_\gamma \; \}^1_{\beta,1} \qquad \{^2_{\beta,1} \; \{^2_\gamma \; \}^2_\gamma \; \}^2_{\beta,1}$$

$$\{^1_\gamma \; \}^1_\gamma \qquad \{^2_\gamma \; \}^2_\gamma$$

The multiple Dyck language $\mathcal{D}$

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \dots, \sigma_k\} \in P$, u_i, v_i well-bracketed:

$$u_0 \sigma_1 v_1 \bar{\sigma}_1 u_1 \cdots \sigma_k v_k \bar{\sigma}_k u_k \equiv_P u_0 \cdots u_k \quad \text{if } v_1 \cdots v_k \equiv_P \varepsilon$$

<u>Example:</u>

$$\{^1_\beta \; \{^1_{\beta,1} \; \{^1_\gamma \; \}^1_\gamma \; \}^1_{\beta,1} \; \}^1_\beta \; \{^2_\beta \; \{^2_{\beta,1} \; \{^2_\gamma \; \}^2_\gamma \; \}^2_{\beta,1} \; \}^2_\beta$$

$$\{^1_{\beta,1} \; \{^1_\gamma \; \}^1_\gamma \; \}^1_{\beta,1} \qquad \{^2_{\beta,1} \; \{^2_\gamma \; \}^2_\gamma \; \}^2_{\beta,1}$$

$$\{^1_\gamma \; \}^1_\gamma \qquad \{^2_\gamma \; \}^2_\gamma \qquad \equiv_P \varepsilon$$

The multiple Dyck language D

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \dots, \sigma_k\} \in P$, u_i, v_i well-bracketed:

$$u_0 \sigma_1 v_1 \bar{\sigma}_1 u_1 \cdots \sigma_k v_k \bar{\sigma}_k u_k \equiv_P u_0 \cdots u_k \quad \text{if} \quad v_1 \cdots v_k \equiv_P \varepsilon$$

Example:

$$\{_\beta^1 \; \{_{\beta,1}^1 \; \{_\gamma^1 \; \}_\gamma^1 \; \}_{\beta,1}^1 \; \}_\beta^1 \; \{_\beta^2 \; \{_{\beta,1}^2 \; \{_\gamma^2 \; \}_\gamma^2 \; \}_{\beta,1}^2 \; \}_\beta^2$$

$$\{_{\beta,1}^1 \; \{_\gamma^1 \; \}_\gamma^1 \; \}_{\beta,1}^1 \qquad \{_{\beta,1}^2 \; \{_\gamma^2 \; \}_\gamma^2 \; \}_{\beta,1}^2 \qquad \equiv_P \varepsilon$$

$$\{_\gamma^1 \; \}_\gamma^1 \qquad \{_\gamma^2 \; \}_\gamma^2 \qquad \equiv_P \varepsilon$$

The multiple Dyck language D

given by an equivalence relation $\equiv_P$

$\forall \{\sigma_1, \dots, \sigma_k\} \in P$, u_i, v_i well-bracketed:

$$u_0 \sigma_1 v_1 \bar{\sigma}_1 u_1 \cdots \sigma_k v_k \bar{\sigma}_k u_k \equiv_P u_0 \cdots u_k \quad \text{if} \quad v_1 \cdots v_k \equiv_P \varepsilon$$

Example:

$$\{^1_\beta \ \{^1_{\beta,1} \ \{^1_\gamma \ \}^1_\gamma \ \}^1_{\beta,1} \ \}^1_\beta \ \{^2_\beta \ \{^2_{\beta,1} \ \{^2_\gamma \ \}^2_\gamma \ \}^2_{\beta,1} \ \}^2_\beta \equiv_P \varepsilon$$

$$\{^1_{\beta,1} \ \{^1_\gamma \ \}^1_\gamma \ \}^1_{\beta,1} \ \{^2_{\beta,1} \ \{^2_\gamma \ \}^2_\gamma \ \}^2_{\beta,1} \equiv_P \varepsilon$$

$$\{^1_\gamma \ \}^1_\gamma \ \{^2_\gamma \ \}^2_\gamma \equiv_P \varepsilon$$

Using the CS representation for n-best parsing

$$\text{"}L = h(R \cap D)\text{"}$$

with our construction:

- R encodes local properties of L
- D filters out elements of R that are not well-formed
- h removes the "{" and "}" symbols
- $R \cap D$ encodes derivation trees:

$$\begin{tikzcd}
R \cap D \subseteq \Delta^* \arrow[d, "h"'] \arrow[dr, "\mathrm{toDeriv}"] & \\
L \subseteq \Sigma^* \arrow[r, bend left, "\mathrm{parse}"] & T_R \cong D_G \arrow[l, bend left, "\mathrm{yield}"]
\end{tikzcd}$$

The parser

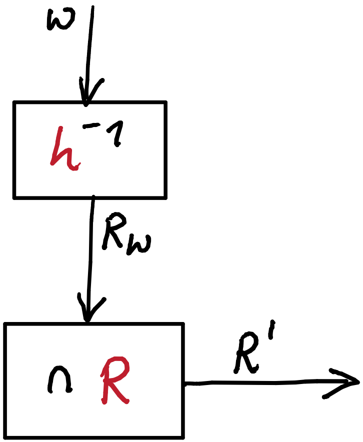
$$\begin{tikzcd}
R\cap D \subseteq \Delta^* \arrow[d, "h"'] \arrow[dr, "\mathrm{toDeriv}"] & \\
L \subseteq \Sigma^* \arrow[r, bend left, "\mathrm{parse}"] & T_R \cong D_G \arrow[l, bend left, "\mathrm{yield}"]
\end{tikzcd}$$

The parser

$$
\begin{tikzcd}
{} \arrow[d, "w"'] \\
\boxed{h^{-1}} \arrow[d, "R_w"] \\
{}
\end{tikzcd}
$$

$$
\begin{tikzcd}
R \cap D \subseteq \Delta^* \arrow[d, "h"'] \arrow[dr, "\mathrm{toDeriv}"] & \\
L \subseteq \Sigma^* \arrow[r, bend left, "\mathrm{parse}"] & T_R \supseteq D_G \arrow[l, bend left, "\mathrm{yield}"]
\end{tikzcd}
$$

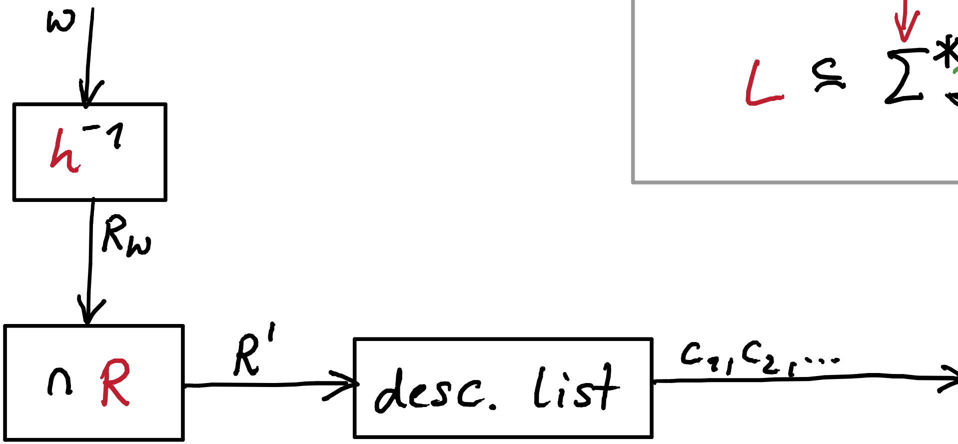
The parser



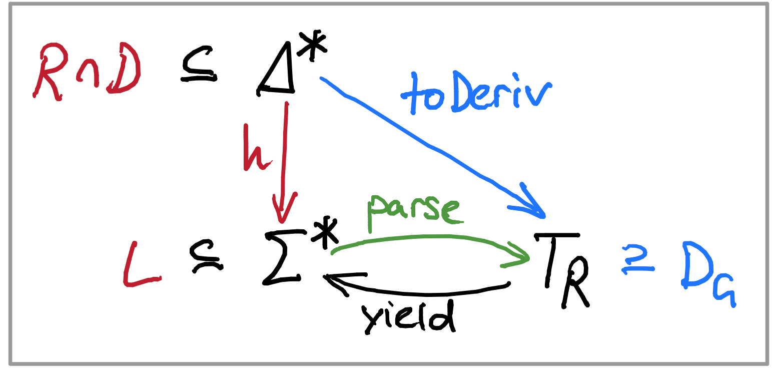
$$
\begin{tikzcd}
R \cap D \subseteq & \Delta^* \arrow[d, "h"'] \arrow[drr, "\mathrm{toDeriv}"] & & & \\
L \subseteq & \Sigma^* \arrow[rr, bend left, "\mathrm{parse}"] & & T_R \arrow[ll, bend left, "\mathrm{yield}"] & \supseteq D_G
\end{tikzcd}
$$

The parser

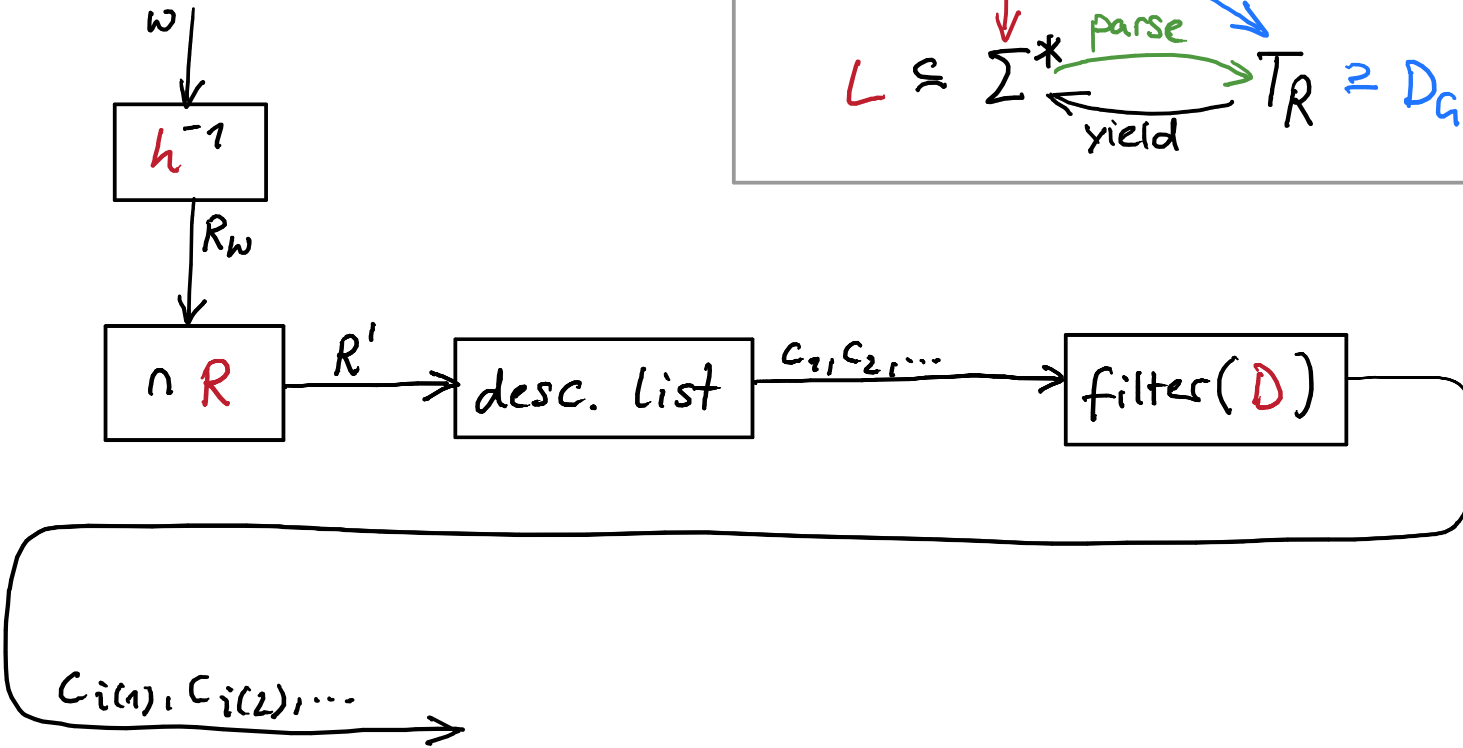
$$
\begin{tikzcd}
R \cap D \subseteq & \Delta^* \arrow[d, "h"] \arrow[dr, "\mathrm{toDeriv}"] & \\
L \subseteq & \Sigma^* \arrow[r, bend left, "\mathrm{parse}"] & T_R \cong D_G \arrow[l, bend left, "\mathrm{yield}"]
\end{tikzcd}
$$



The parser

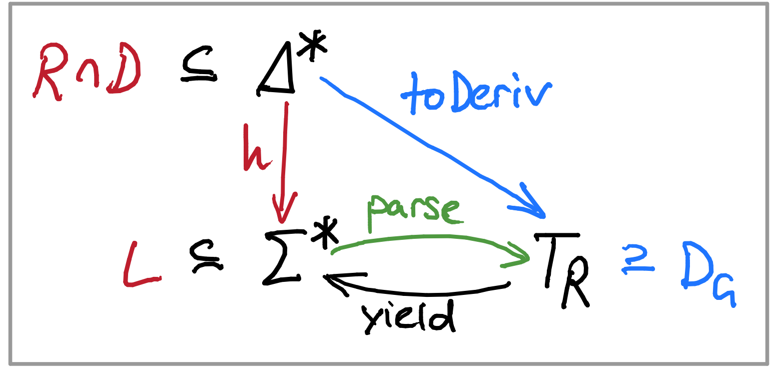


$R \cap D \subseteq \Delta^*$
h
toDeriv
parse
$L \subseteq \Sigma^*$
$T_R \supseteq D_G$
yield

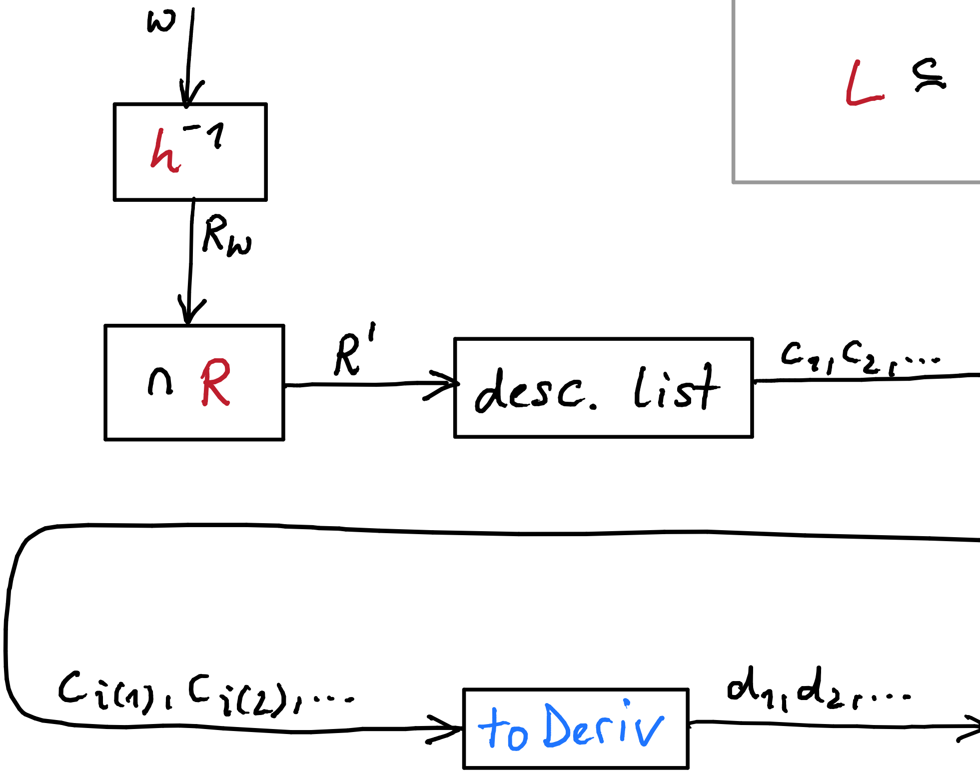


w
h^{-1}
R_w
$\cap R$
R'
desc. list
$c_1, c_2, \ldots$
filter(D)
$c_{i(1)}, c_{i(2)}, \ldots$

The parser

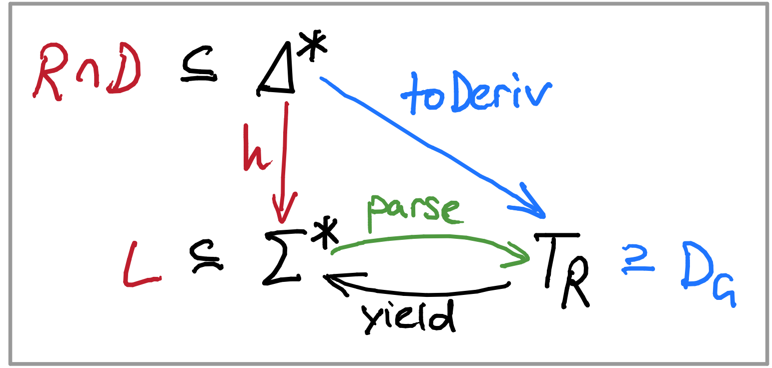


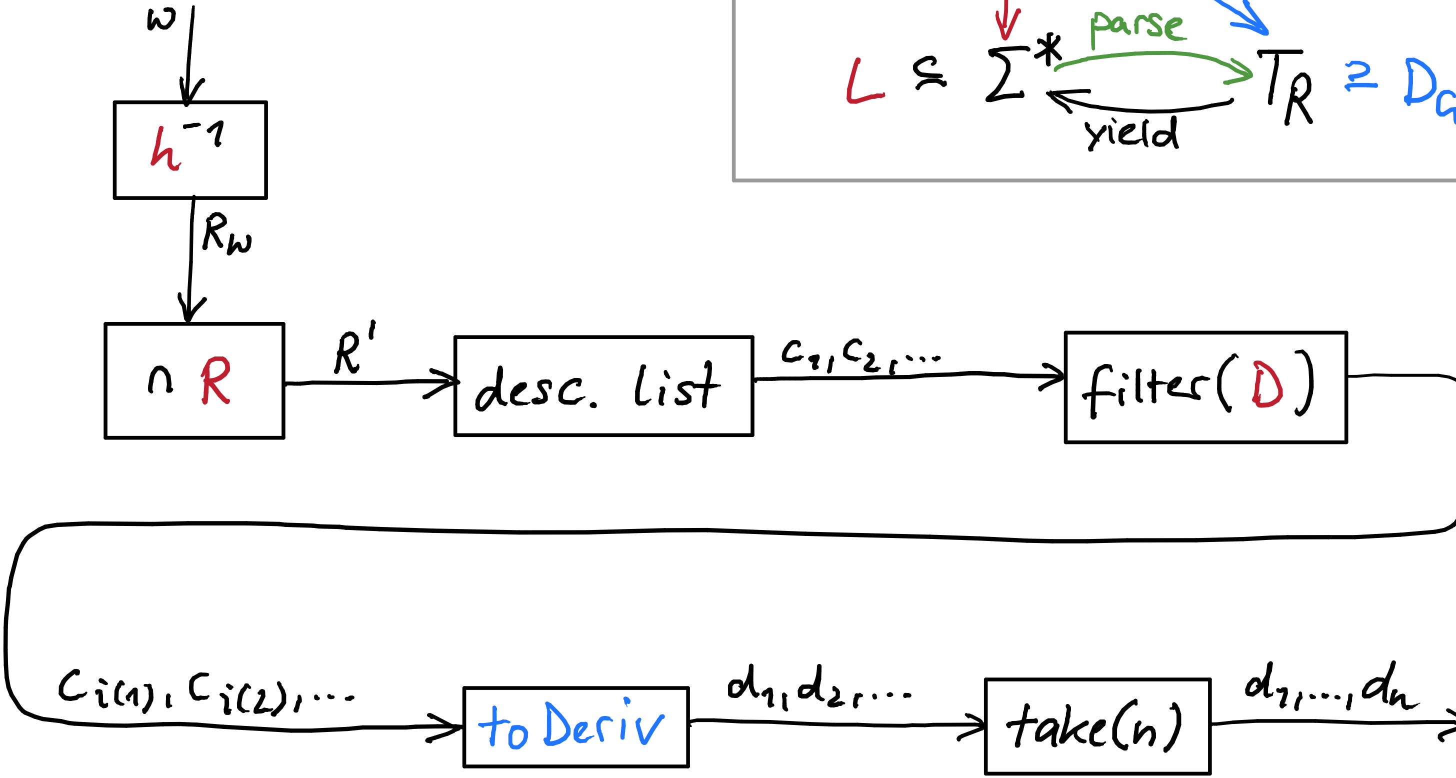
R∩D ⊆ Δ*
toDeriv
h
parse
L ⊆ Σ*
T_R ≅ D_G
yield



w
h⁻¹
R_w
∩ R
R'
desc. list
c₁, c₂, ...
filter(D)
c_i(1), c_i(2), ...
to Deriv
d₁, d₂, ...

The parser


$R \cap D \subseteq \Delta^*$
toDeriv
h
parse
$L \subseteq \Sigma^*$
$T_R \supseteq D_G$
yield


w
h^{-1}
R_w
$\cap R$
R'
desc. list
$c_1, c_2, \ldots$
filter(D)
$c_{i(1)}, c_{i(2)}, \ldots$
toDeriv
$d_1, d_2, \ldots$
take(n)
$d_1, \ldots, d_n$

Further research (to be done by Thomas Ruprecht)

- implement the parser in rustomata*
  - use OpenFST for $\boxed{\cap R}$ and $\boxed{\text{desc. list}}$
  - implement $\boxed{\text{filter}(D)}$ as an automaton (maybe)
- optimise the parser
  - pruning
  - tweaking the automata R and R_w
  - approximate D and incorporate into R
- evaluation

*https://github.com/tud-fop/rustomata