Computational Linguistics II: Parsing

The CYK Parser

Frank Richter & Jan-Philipp Söhn

fr@sfs.uni-tuebingen.de, jp.soehn@uni-tuebingen.de

Inefficiencies of TD and BUP Parsers

- Attachment ambiguities lead to ‘reparsing’ of partial, correct structures

Inefficiencies of TD and BUP Parsers

- Attachment ambiguities lead to ‘reparsing’ of partial, correct structures
- Consequence: inefficient behavior of the algorithms

Inefficiencies of TD and BUP Parsers

- Attachment ambiguities lead to ‘reparsing’ of partial, correct structures
- Consequence: inefficient behavior of the algorithms
- In order to avoid ‘reparsing’, a parser needs to remember partial results.

Cocke-Younger-Kasami Parser – CYK

- The CYK parser uses chart for storing partial results.

Cocke-Younger-Kasami Parser – CYK

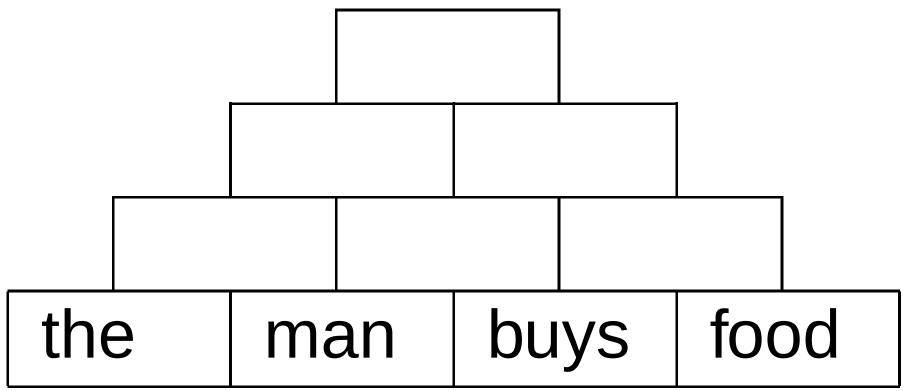
- The CYK parser uses chart for storing partial results.
- It requires grammar in Chomsky Normal Form.

Cocke-Younger-Kasami Parser – CYK

- The CYK parser uses chart for storing partial results.
- It requires grammar in Chomsky Normal Form.
- The chart takes the form of a (partial) matrix.

Cocke-Younger-Kasami Parser – CYK

- The CYK parser uses chart for storing partial results.
- It requires grammar in Chomsky Normal Form.
- The chart takes the form of a (partial) matrix.



A few Definitions

Definition 1

Two context free grammars G and G' are called *equivalent* iff $L(G) = L(G')$.

Definition 2

A context free grammar $G = \langle N, T, P, S \rangle$ is called ϵ-free if it does not contain a production of the form $A \longrightarrow \epsilon$, with the possible exception of $S \longrightarrow \epsilon$ in case $\epsilon \in L(G)$. In this case, S does not occur on the righthand side of a production rule in P.

Definition 3

For each grammar $G = \langle N, T, P, S \rangle$, each $p \in P$ of the form $A \longrightarrow B$ with $A, B \in N$, is called a *chain rule* (*unit rule*).

ϵ-free Type 2 Grammars (1)

Lemma 1

There is an algorithm which for each context free grammar $G = \langle N, T, P, S \rangle$ produces an equivalent ϵ-free context free grammar $G' = \langle N', T', P', S' \rangle$.

Sketch of the procedure: Let
$W_1 = \{A \in N | A \rightarrow \epsilon \in P\}$,
$W_{i+1} = \{A \in N | A \rightarrow x \in P$ with $x \in W_i^*\} \cup W_i$.

1. $W_i \subseteq W_{i+1}$ $(i \geq 1)$,
2. If $W_i = W_{i+1}$ then $W_i = W_{i+m}$ for $m \geq 0$,
3. $W_n = W_{n+1}$, $n = |N|$,
4. $W_n = \{A \in N | A \rightarrow^* \epsilon\}$

ϵ-free Type 2 Grammars (2)

Let
$N' = N \cup S'$, S' a new start symbol,
$T' = T$,
$P' = \{S' \rightarrow S\} \cup$
$\{A \rightarrow A_1 \ldots A_k | k \geq 1, A_i \in N \cup T$
and there are $x_1 \ldots x_{k+1} \in W_n^*$ with
$A \rightarrow x_1 A_1 x_2 \ldots A_k x_{k+1} \in P\} \cup P_\epsilon$,
with $P_\epsilon = \{\}$ in case $\epsilon \notin L(G)$, else $P_\epsilon = S' \rightarrow \epsilon$.