**Parsing**

**Weighted Deductive Parsing**

Laura Kallmeyer  
Heinrich-Heine-Universität Düsseldorf  
Wintersemester 2012/2013

---

**Overview**

1. Idea  
2. Algorithm  
3. Example  
4. Parsing

---

**Idea (1)**

Idea of weighted deductive parsing [?]:

- Give a deductive definition of the probability of a parse tree.
- Use Knuth’s algorithm to compute the best parse tree for category $S$ and a given input $w$.

Advantage:

- Yields the best parse without exhaustive parsing.
- Can be used to parse any grammar formalism as long as an appropriate weighted deductive system can be defined.

---

**Idea (2)**

Parsing Schemata understand parsing as a deductive process.

Deduction of new items from existing ones can be described using inference rules.

General form:

$$\text{antecedent} \quad \text{side conditions} \quad \text{consequent}$$

antecedent, consequent: lists of items

Application: if antecedent can be deduced and side condition holds, then the consequent can be deduced as well.
Idea (3)
A parsing schema consists of
• deduction rules;
• an axiom (or axioms), can be written as a deduction rule with empty antecedent;
• and a goal item.
The parsing algorithm succeeds if, for a given input, it is possible to deduce the goal item.

Deduction-based definition of bottom-up CFG parsing (CYK) with Chomsky Normal Form.
For an input \( w = w_1 \cdots w_n \) with \( |w| = n \),
1. Item form \([A, i, j]\) with \( A \) a non-terminal, \( 1 \leq i \leq j \leq n \).
2. Deduction rules:
   Scan: \( [A, i, i] \) \( A \rightarrow w_i \)
   Complete: \( [B, i, j], [C, j + 1, k] \) \( A \rightarrow B C \)
3. Goal item: \([S, 1, n]\).

Rule applications:

\[
\begin{array}{c}
\text{Axioms (Scan):} \\
\text{Complete:}
\end{array}
\]

\[
\begin{array}{c}
\text{Scan:} \\
\text{Complete:}
\end{array}
\]

Extending CYK with weights:

\[
\begin{array}{c}
\text{Scan:} \\
\text{Complete:}
\end{array}
\]

Idea (5)

Example: \textit{sees the man}

Idea (6)

Extension to a weighted deduction system:
• Each item has an additional weight. Intuition: weight = costs to build an item.
• The deduction rules specify how to compute the weight of the consequent item form the weights of the antecedent items.
Algorithm (1)

- There is a linear order $<$ defined on the weights.
- The lower the weight, the better the item.
- For Knuth’s algorithm, the weight functions $f$ must be monotone nondecreasing in each variable and $f(x_1, \ldots, x_m) \geq \max(x_1, \ldots, x_m)$.

In our example, this is the case:

Complete: $x_1: [B, i, j], x_2: [C, j + 1, k]$  
$p: A \rightarrow B C$  
$f(x_1, x_2) = x_1 + x_2 + c$ where $c \geq 0$ is a constant.

Algorithm (2)

Algorithm for computing the goal item with the lowest weight, goes back to Knuth.

Goal: Find possible items with their lowest possible weight.

We need two sets:

- A set $C$ (the chart) that contains items that have reached their final weight.
- A set $A$ (the agenda) that contains items that are waiting to be processed as possible antecedents in further rule applications and that have not necessarily reached their final weight.

Initially, $A = \emptyset$ and $A$ contains all items that can be deduced from an empty antecedent set.

Algorithm (3)

while $A \neq \emptyset$ do
   remove the best item $x : I$ from $A$ and add it to $C$
   if $I$ goal item
      then stop and output true
   else
      for all $y : I'$ deduced from $x : I$ and items in $C$
         if there is no $z$ with $z : I' \in C$ or $z : I' \in A$
            then add $y : I'$ to $A$
         else if $z : I' \in A$ for some $z$
            then update weight of $I'$ in $A$ to $\min(y, z)$

Algorithm (4)

If the weight functions are as required, then the following is guaranteed:

- Whenever an item is the best in the agenda, you have found its lowest weight.
- Therefore, if this item is a goal item, then you have found the best parse tree for your input.
- If it is no goal item, you can store it in the chart.

⇒ no exhaustive parsing needed.

However, $A$ needs to be treated as a priority queue which can be expensive.
**Example**

\[ S \rightarrow SS \]
\[ S \rightarrow a \]
\[ A \rightarrow a \]
\[ A \rightarrow b \]

Input: aa

<table>
<thead>
<tr>
<th>Chart</th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 : [A, 1, 1]</td>
<td>4 : [A, 2, 2], 5 : [S, 1, 1], 5 : [S, 2, 2]</td>
</tr>
<tr>
<td>4 : [A, 1, 2]</td>
<td>4 : [A, 2, 2], 5 : [S, 1, 1], 5 : [S, 2, 2]</td>
</tr>
<tr>
<td>4 : [A, 1, 1], 4 : [A, 2, 2]</td>
<td>5 : [S, 1, 1], 5 : [S, 2, 2]</td>
</tr>
<tr>
<td>4 : [A, 1, 1], 4 : [A, 2, 2], 5 : [S, 1, 1]</td>
<td>5 : [S, 2, 2], 1 : [S, 1, 2]</td>
</tr>
<tr>
<td>4 : [A, 1, 1], 4 : [A, 2, 2], 5 : [S, 1, 1]</td>
<td>1 : [S, 1, 2]</td>
</tr>
<tr>
<td>5 : [S, 2, 2]</td>
<td>1 : [S, 1, 2]</td>
</tr>
</tbody>
</table>

**Parsing**

Extension to parsing:

- Whenever we generate a new item, we store it not only with its weight but also with backpointers to its antecedent items.
- Whenever we update the weight of an item, we also have to update the backpointers.

In order to read off the best parse tree, we have to start from the best goal item and follow the backpointers.