Chart Parsing

- Parsing: Efficiency Issues
- Chart Parsing
  - Charts
  - Edges
  - Rules
- Chart Parsing in Prolog
- Demonstration

Parsing: Efficiency Issues

- Problems with simple top-down parsing:
  - "Left-recursive" rules can cause infinite loops
    - NP → NP and NP
  - Redundant parsing of phrases.
    - "I saw the dog in the tall building behind the hill."
      (the dog was in the building)
    - "I saw the dog in the tall building behind the hill."
      (I was in the building)

Parsing: Efficiency Issues (2)

- Problems with simple bottom-up parsing:
  - Builds structures that are locally valid but not useful globally.
- Solutions:
  - Re-use the sub-parses we've already computed
  - Combine top-down and bottom-up approaches
    - Get the "best of both worlds"
    - We need some common representation for the information from top-down and bottom-up approaches.
    - Use heuristics to decide when to use bottom-up or top-down approaches.

Chart Parsing

- Use a chart to record hypotheses about possible syntactic constituents.
  - A chart contains a set of edges.
- Each edge represents a possible phrase.
  - Edges provide a common representation for parse information.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Phrase Type</th>
<th>Phrase Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>I saw the man on the hill.</td>
<td>NP</td>
<td>Det N</td>
</tr>
</tbody>
</table>
Edges

Edges can represent partial phrases.

<table>
<thead>
<tr>
<th></th>
<th>saw</th>
<th>the</th>
<th>man</th>
<th>on</th>
<th>the</th>
<th>hill</th>
</tr>
</thead>
</table>

PP starts here
So far, we’ve found a P
We still need an NP

Edges (continued)

• An edge consists of:
  - S: A start index (1...n)
  - E: An end index (1...n)
  - Type: A phrase type (NP, PP, etc.)
  - Found: What we’ve found so far (list of phrase types)
  - Need: What we still need (list of phrase types)

• Representing edges in Prolog:
  - edge(S, E, Type, Found, Need)

Chart Parser Rules

• A chart parser rule adds new edges to the chart.
• Each chart parsing strategy defines a set of rules.
  - Top down:
    • top-down initialization rule
    • top-down rule
    • fundamental rule
  - Bottom-up:
    • bottom-up rule
    • fundamental rule

The Fundamental Rule

• The fundamental rule is used by both top-down and bottom-up strategies.

<table>
<thead>
<tr>
<th>If the chart contains:</th>
<th>Then add:</th>
</tr>
</thead>
</table>
|\[
\begin{array}{c}
A \\
\alpha \\
C \\
\end{array}
\begin{array}{c}
C \\
\beta \\
\end{array}
\] |\[
\begin{array}{c}
A \\
\alpha C \\
\end{array}
\begin{array}{c}
C \\
\beta \\
\end{array}
\] |
**Top-Down Rules**

- **Top-down initialization:**
  
  For any rule $S \rightarrow \alpha$:
  - Add $S \rightarrow \alpha$ to the left side of the chart (start=end=1).

- **Top-down rule:**

  For each rule $X \rightarrow Y\beta$ and $Y\gamma$:
  - Add $X \rightarrow Y\beta\gamma$ to the left side of the chart.

**Bottom-Up Rules**

- **Bottom-Up Rule**

  If the chart contains: $A \rightarrow B\alpha\beta$
  
  For each rule $B \rightarrow A\beta$
  - Add $B \rightarrow A\alpha\beta$ to the right side of the chart.

**Chart Parsing in Prolog**

- **Define edges using a relation:**
  - `edge(S, E, Type, Found, Need, ParseTree)`

- **Define resulting parse trees using a structure:**
  - `parsetree(Type, ChildList)`

- **Define rules as conditions on the edge relation.**
  - Example: Fundamental rule
    ```prolog
    edge(X, Y, A, BC, [], parsetree(A,PT1,PT2)) :-
    edge(X, Y, A, B, [C], PT1),
    edge(Y, Z, C, D, [], PT2),
    append(B, [C], BC).
    ```

**Chart Parsing Demo...**