

*October 19, 2004*  
*Chapter 10.3–10.6*  
*Parsing*

## *Overview*

- Review: CFGs, basic top-down parser
- Dynamic programming
- Earley algorithm (how it works, how it solves the problems)
- Finite-state parsing

## *Last time*

- What is parsing?
- Inadequate grammars for human languages
  - lists,
  - regex,
  - CFG
- A simple top-down parser for CFGs

## *Corrected version of TOP-DOWN-PARSE*

```
function TOP-DOWN-PARSE(input, grammar) returns a parse tree
  agenda  $\leftarrow$  (Initial S tree, Beginning of input)
  css  $\leftarrow$  POP(agenda)
  loop
    if SUCCESSFUL-PARSE?(css) then
      return TREE(css)
    else
      if CAT(NODE-TO-EXPAND(css)) is a POS then
        if CAT(NODE-TO-EXPAND(css))  $\in$ 
          POS(CURRENT-INPUT(css)) then
          PUSH(APPLY-LEXICAL-RULE(css, grammar, agenda))
        else
          PUSH(APPLY-RULES(css, grammar, agenda))
      if agenda is empty then
        return reject
      else
        css  $\leftarrow$  POP(agenda)
  end
```

## *Problems with* TOP-DOWN-PARSE

- Infinite loops with left-recursive grammars
- Ambiguity
- Inefficient reparsing of subtrees

# *Dynamic Programming*

- First introduced by Bellman (1957)
- A table-driven method of solving problems by combining solutions to sub-problems
- In this context, what is the problem?
- What is a sub-problem?
- How are sub-problems combined?

## *The Earley Chart Parser: Chart and Dotted Rules*

- For a sentence of  $N$  words, the chart contains  $N+1$  cells.
- Each cell contains a list of states.
- A state consists of: a local subtree (‘edge’), information about the degree of completion of that subtree, and information about how much of the string corresponds to the subtree.
- For example (in dotted-rule notation):
  - $S \rightarrow \bullet VP, [0, 0]$
  - $NP \rightarrow Det \bullet Nominal, [1, 2]$
  - $VP \rightarrow VNP\bullet, [0, 3]$

## *The Earley Chart Parser, Outline*

- Add the initial S ( $\textit{gamma} \rightarrow \bullet S, [0,0]$ ) to the chart at position 0.
- Loop, for each of the rest of the cells in the chart:
  - If the state is incomplete, and the category to the right of the dot is not a POS, add (if not redundant) new states to the chart in the current position for each rule that expands that category. These new states all have the dot at the beginning of the rule, and a span that starts and ends at the current position. (PREDICTOR)



## *The Earley Chart Parser, Outline*

- Loop (continued):
  - If the state is incomplete, and the category to the right of the dot ('B') is a POS, and B is a possible POS for the next word in the string, add the rule  $B \rightarrow word$  ● (if not redundant) to the next cell in the chart. (SCANNER)
  - If the state is complete (dot all the way to the right), look in the cell corresponding to the beginning of the current state's span for states which are currently seeking a daughter of the same category as the mother of this state. For each one of those, add a state (if not redundant) to the *current* cell, with the dot moved over one, and the span increased to the end of the current word. (COMPLETER)

## *The Earley Chart Parser*

- In which cell of the chart does one find the spanning edge(s)?
- Is the Earley algorithm top-down or bottom-up?
- Best-first or exhaustive?
- Uni-directional or bi-directional?
- Breadth-first or depth-first?

*How does the Earley algorithm solve these problems?*

- Ambiguity
- Inefficient reparsing of subtrees
- Infinite loops with left-recursive grammars
  - $\text{NP} \rightarrow \text{NP PP}$
  - *The cat on the mat by the door slept*

## *Expanding the Chart*

- Parsing is apparently an exponential-time problem.
- The Earley algorithm does *recognition* in polynomial time ( $O(N^3)$ ).
- Returning the trees is still potentially exponential.
- How would this algorithm return the trees?
- Effectively takes an exponential-time problem and makes it polynomial.

## *Finite-State or ‘Chunk’ Parsing*

- When robustness is more important than precision, *shallow* parsing techniques are used instead.
- Finite-State or ‘Chunk’ parsers recognize patterns within sentences as ‘Noun groups’, ‘Verb groups’ etc.
- They can be used to return partial results, even if the whole string can’t be treated.
- Done with cascades of FSAs. What does it mean to ‘cascade’ an FSA?

## *Finite-State or ‘Chunk’ Parsing*

- More modern robust, shallow processing techniques involve machine learning to deal with learning all the different patterns.
- One example: RASP (Robust Accurate Statistical Parsing)  
<http://www.cogs.susx.ac.uk/lab/nlp/rasp>
- The European project Deep Thought is currently investigating methods for combining deep and shallow parsing for knowledge-intensive information extraction  
<http://www.project-deepthought.net/>

## *Overview*

- Review: CFGs, basic top-down parser
- Dynamic programming
- Earley algorithm (how it works, how it solves the problems)
- Finite-state parsing

## *Coming up next...*

- Adding unification to enable more interesting grammars.
- Another parsing algorithm, with a different kind of chart.
- Probabilistic parsing, for meaningful best-first
- A parser such as the Earley Chart Parser is only as good as the grammars it interprets...