Computation, Information, and Intelligence (COMS/ENGRI/INFO/COGST 172), Fall 2005
10/26/05: **Lecture 26 aid — Earley parsing**

---

**Agenda**: an algorithm for recovering the analyses of an input sentence with respect to a given context-free grammar (CFG).

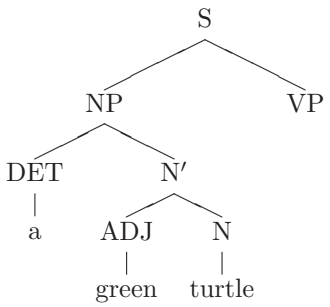
**Announcements**:

- The second prelim will be held in class on Friday, November 4th; topic coverage: IR and Web-based retrieval. Same drill as before (including the fact that no homework will be assigned before then), except that I don't anticipate being able to provide additional sample problems. Further details, including changes to the usual office hour schedule, are forthcoming.
- If you are either in favor or against changing the Monday 8pm regular office hour to Monday 1:30-2:30, please let me know by Thursday midnight by email (stating the reasons for your preference, e.g., conflict with the changed time but not the original time, or vice versa).

## I. Garden-path sentences

1. "The cotton clothing is made of grows in Mississippi."
2. "The player kicked the ball kicked the ball."

**II. Example Earley-style partial parse trees** The sentence being parsed is "a green turtle swam to shore".




The point is that the leftmost leaves correspond to the first words of the sentence being parsed: Earley's algorithm tries to "guess" at how the parse tree will grow, but "anchors" its guesses against the words of the sentence in a left-to-right manner.

**III. Parse states** Assume a fixed sentence $w_1 w_2 \ldots w_n$ and CFG with start non-terminal S. The general form of a parse state is

$$(X \to \alpha \bullet \beta, i, j),$$

where

- $\alpha$ and $\beta$ are some "stuff" (a sequence of zero or more terminals or nonterminals) such that $X \to \alpha beta$ is a rewrite rule in the CFG,
- $i$ and $j$ range between 0 and $n$, and are *usually* interpreted as indicated endpoints of some subsequence of the sentence, and
- if $1 \leq i \leq j$, then we have inferred through the parsing process that $\alpha$ (i.e., the "stuff" before the "dot") can be rewritten into the sentence subsequence $w_i w_{i+1} \cdots w_j$.

**IV. Parser actions** As above, we use Greek lower case letters to indicate "stuff" (a sequence of zero or more terminals or nonterminals). So, if we have rewrite rule $A \to BcDD$, we are allowed to describe it as $A \to \alpha B \beta$ or $A \to \alpha BcDD$ (the case in which $\alpha$ is a sequence of zero symbols) or a multitude of other ways.

1. *Predict* (guess how to expand the leftmost nonterminal not yet known to account for any part of the sentence):

   Given $(X \to \alpha \bullet Y\beta, i, j)$ and rewrite rule $Y \to \gamma$, add state $(Y \to \bullet\gamma, j+1, j)$.

   Note the special treatment of what are usually endpoint indicators, meant to remind us that we don't know how far into the sentence that $\gamma$ will "cover".

2. *Scan* (advance the "dot" over a terminal that matches the next word in the sentence):

   Given $(X \to \alpha \bullet y\beta, i, j)$ and the fact that $y = w_{j+1}$, add state $(X \to \alpha y \bullet \beta, i, j+1)$.

3. *Complete* (advance the "dot" over a nonterminal that we now know can be rewritten into the next subsequence of words in the sentence):

   Given $(X \to \alpha \bullet Y\beta, i, j)$ and $(Y \to \gamma\bullet, j+1, k)$, add state $(X \to \alpha Y \bullet \beta, i, k)$.

**V. Earley's algorithm** This algorithm is guaranteed to terminate.

1. Start with the special state ( $\to \bullet S, 1, 0$).
2. Perform all possible predictions
3. Scan.
4. Perform all possible completions.
5. Return to step 2 unless nothing changed from the previous round.

**VI. Example partial execution** Note that the process would be more interesting for grammars generating ambiguous output.

For brevity (don't do this at home!), we just give the rewrite rules for a sample CFG with start nonterminal S:

| | |
|---|---|
| (1) S → NP VP | (2) ADJ → green |
| (3) NP → DET N′ | (4) N → turtle |
| (5) N′ → ADJ N | (6) V → swam |
| (7) VP → V P N | (8) P → to |
| (9) DET → a | (10) N → shore |

Sentence: "a green turtle swam to shore".

| | | | | | |
|---|---|---|---|---|---|
| *init*: | ( → • S, 1, 0) | *predict*: | (N′ →•ADJ N, 2, 1) | *predict*: | (N →•turtle, 3, 2) |
| *predict*: | (S →•NP VP, 1, 0) | | (ADJ →•green, 2, 1) | | (N →•shore, 3, 2) |
| | (NP → • DET N′, 1, 0) | *scan*: | (ADJ →green •, 2, 2) | *scan*: | (N →turtle •, 3, 3) |
| | (DET →•a, 1, 0) | *complete*: | (N′ →ADJ •N, 2, 2) | *complete*: | (N′ →ADJ N •, 2, 3) |
| *scan*: | (DET →a •, 1, 1) | | | | (NP → DET N′ •, 1, 3) |
| *complete*: | (NP → DET • N′, 1, 1) | | | | (S →NP •VP, 1, 3) |

This table implicitly encodes both the partial parse tree given on the reverse of this sheet and an alternative that predicts that "shore" would be the third word, but this alternative is never expanded further.