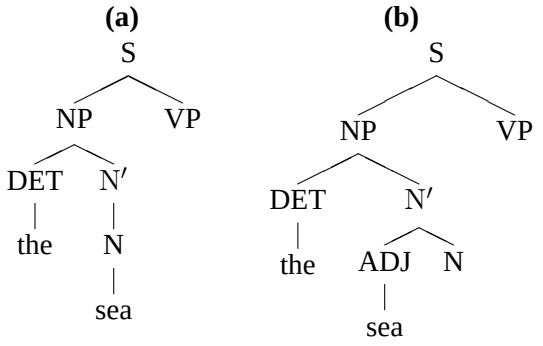
Computation, Information, and Intelligence (ENGRI/CS/INFO/COGST 172), Spring 2007
4/2/07: **Lecture 28 aid — Earley parsing**

Topics: Earley's algorithm for CFG parsing.

Announcements: Ben Pu's office hours on Thursday will definitely be held in Upson 328A. (The schedule handed out on Friday said "probably".)

I. Example Earley-style partial parse trees [1] The sentence being parsed begins with the words, "the sea", and we assume some reasonable CFG for English is being used.



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.

II. 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

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

where

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

[1] These are slightly revised from the aid for last lecture, but we didn't discuss them last time anyway.

III. Parser actions

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

   $$\text{Given } (X \rightarrow \alpha \bullet w_{j+1}\beta, i, j), \text{ add state } (X \rightarrow \alpha w_{j+1} \bullet \beta, i, j+1).$$

2. *Complete* (advance the "dot" over a nonterminal that accounts for the next subsequence of words in the sentence):

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

3. *Predict* (guess how to account for the next part of the sentence):

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

   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 γ will "cover".

IV. Earley's algorithm

This algorithm is guaranteed to terminate.

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

V. Example execution We show only the first few rounds.

For brevity (don't do this at home!), we just give the rewrite rules (rather than all four components) for a CFG with start nonterminal S. Each row corresponds to all the decompositions (branches) for a particular constituent type. The line divides the rules that involve terminals from those that don't (you don't need to do this for any CFGs you write).

(1) S → NP VP		
(2) NP → DET N′		
(3) N′ → ADJ N	(4) N′ → N	
(5) VP → V	(6) VP → V PP	
(7) PP → P N		
(8) ADJ → sea		
(9) DET → the		
(10) N → sea	(11) N → shore	(12) N → turtle
(13) P → to		
(14) V → raged	(15) V → swam	

Sentence: "the sea turtle swam to shore"

the		sea		turtle	
init:	(→•S, 1, 0)	*predict*:	(N′ →•ADJ N, 2, 1)	*predict*:	(N →•sea, 3, 2)
predict:	(S →•NP VP, 1, 0)		(N′ →•N, 2, 1)		(N →•shore, 3, 2)
	(NP →•DET N′, 1, 0)				(N →•turtle, 3, 2)
	(DET →•the, 1, 0)		(ADJ →•sea, 2, 1)		
scan:	(DET →the •, 1, 1)				(VP →•V, 3,2)
complete:	(NP →DET •N′, 1, 1)		(N →•sea, 2, 1)		(V →•raged, 3,2)
			(N →•shore, 2, 1)		(V →•swam, 3,2)
			(N →•turtle, 2, 1)	*scan*:	(N →turtle •, 3, 3)
		scan:	(ADJ →sea •, 2, 2)	*complete*:	(N′ →ADJ N •, 2, 3)
			(N →sea •, 2, 2)		(NP →DET N′ •, 1, 3)
		complete:	(N′ →ADJ •N, 2, 2)		(S →NP •VP, 1, 3)
			(N′ →N •, 2, 2)		
			(NP →DET N′ •, 1, 2)		
			(S →NP •VP, 1, 2)		