UPPSALA UNIVERSITET

The Earley Algorithm

Syntactic analysis (5LN455)

2013-12-04

Sara Stymne
Department of Linguistics and Philology

Based on slides by Marco Kuhlmann

Recap: Treebank grammars, evaluation

Treebanks

- Treebanks are corpora in which each sentence has been annotated with a syntactic analysis.
- Producing a high-quality treebank is both time-consuming and expensive.
- One of the most widely known treebanks is the Penn TreeBank (PTB).

UPPSALA UNIVERSITET

The Penn Treebank

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

Treebank grammars

- Given a treebank, we can construct a grammar by reading rules off the phrase structure trees.
- A treebank grammar will account for all analyses in the treebank.
- It will also account for sentences that were not observed in the treebank.

UPPSALA UNIVERSITET

Treebank grammars

- The simplest way to obtain rule probabilities is relative frequency estimation.
- Step 1: Count the number of occurrences of each rule in the treebank.
- Step 2: Divide this number by the total number of rule occurrences for the same left-hand side.

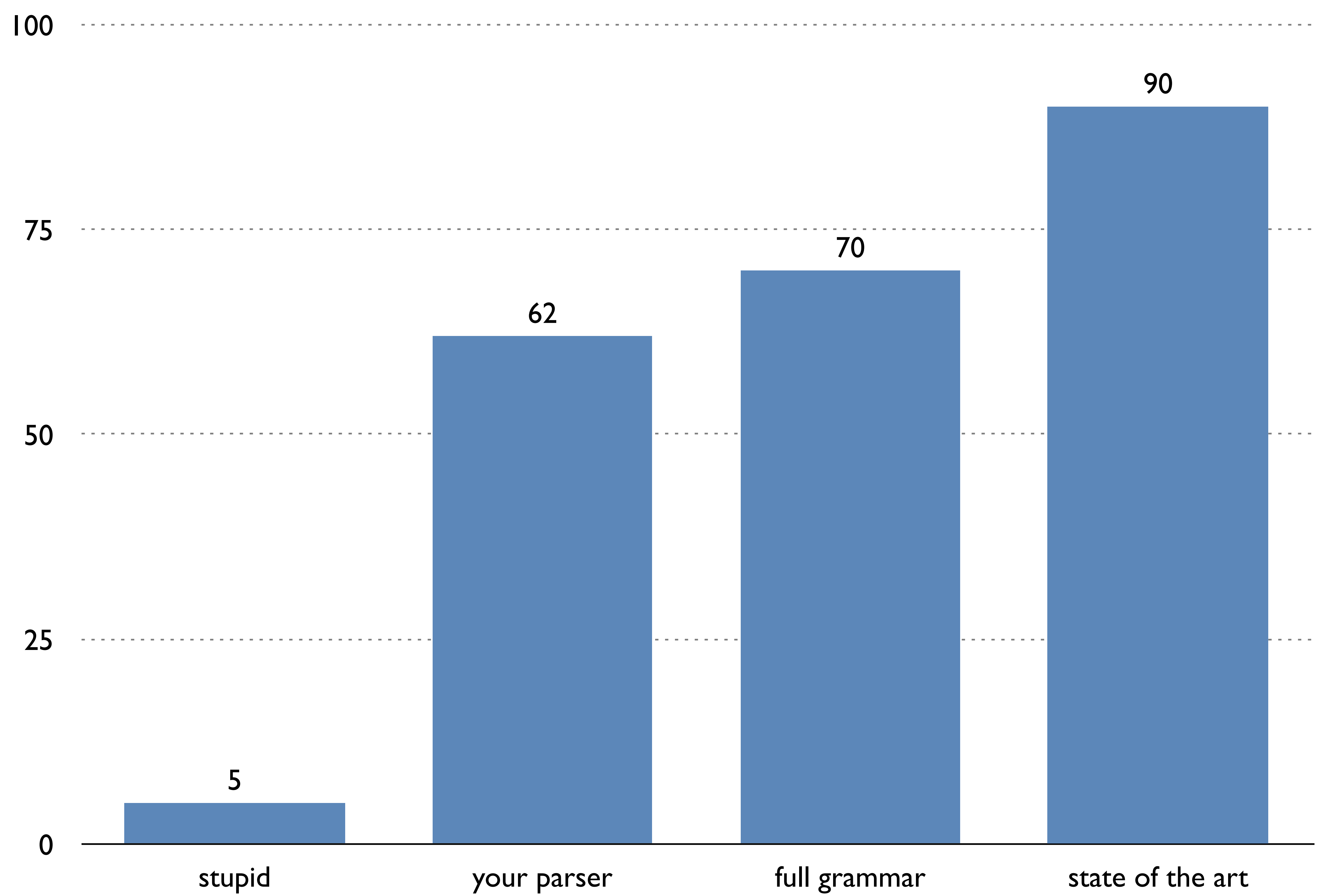
UPPSALA UNIVERSITET

Parse evaluation measures

- Precision:
  Out of all brackets found by the parser,
  how many are also present in the gold standard?
- Recall:
  Out of all brackets in the gold standard,
  how many are also found by the parser?
- F1-score:
  harmonic mean between precision and recall:
  $2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$

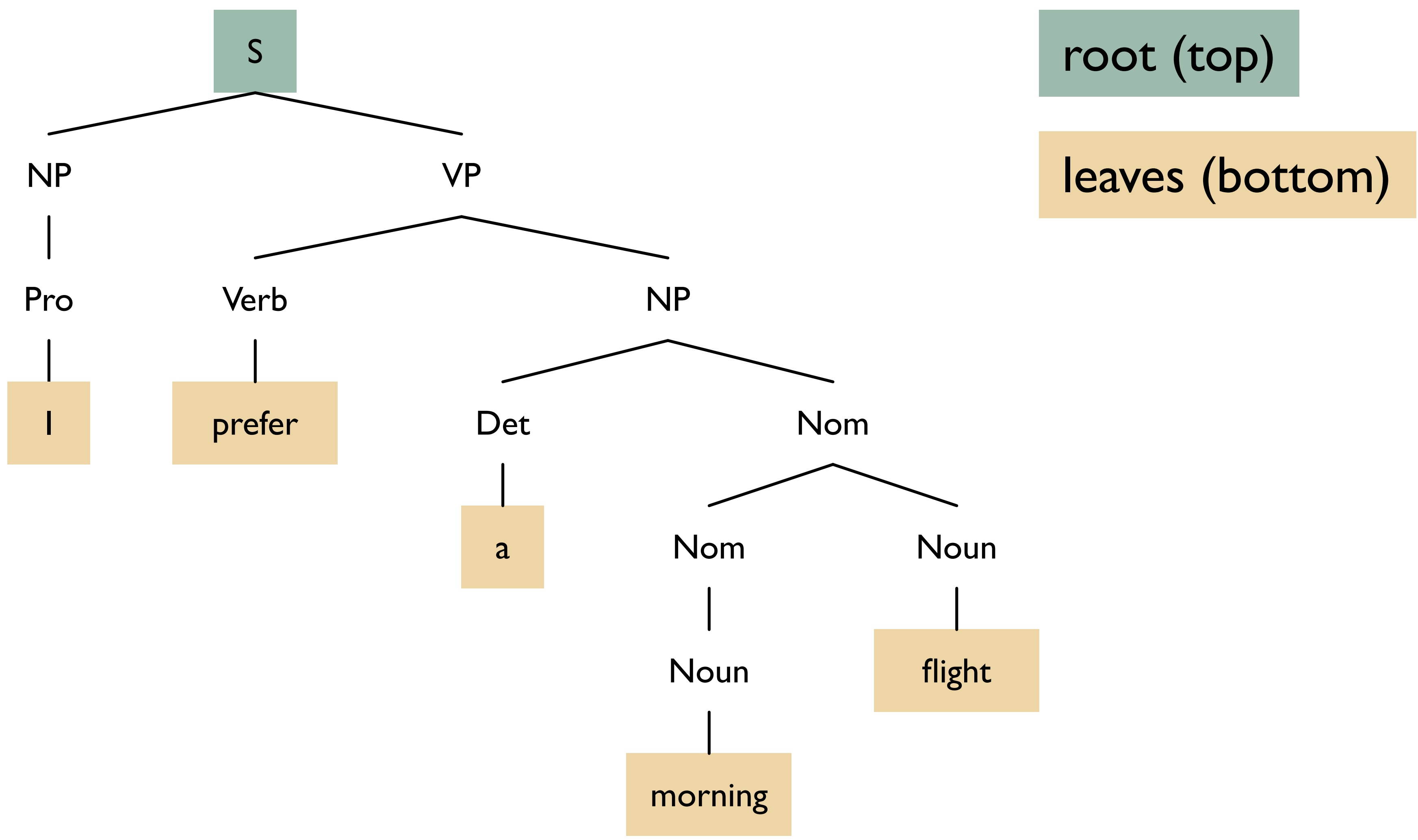
UPPSALA UNIVERSITET

Parser evaluation measures



The Earley algorithm

Parse trees


S
NP
VP
Pro
Verb
NP
I
prefer
Det
Nom
a
Nom
Noun
Noun
flight
morning
root (top)
leaves (bottom)

Top–down and bottom–up

top–down

only build trees that have *S* at the root node

may lead to trees that do not yield the sentence

bottom–up

only build trees that yield the sentence

may lead to trees that do not have *S* at the root

CKY versus Earley

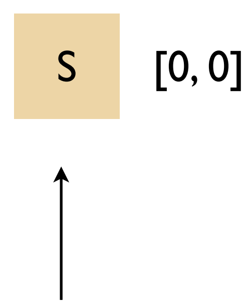
- The CKY algorithm has two disadvantages:
  - It can only handle restricted grammars.
  - It does not use top–down information.
- The Earley algorithm does not have these:
  - It can handle arbitrary grammars.
  - Is does use top–down information.
  - On the downside, it is more complicated.

The algorithm

- Start with the start symbol *S*.
- Take the leftmost nonterminal and predict all possible expansions.
- If the next symbol in the expansion is a word, match it against the input sentence (scan); otherwise, repeat.
- If there is nothing more to expand, the subtree is complete; in this case, continue with the next incomplete subtree.

Example run

0 I 1 prefer 2 a 3 morning 4 flight 5

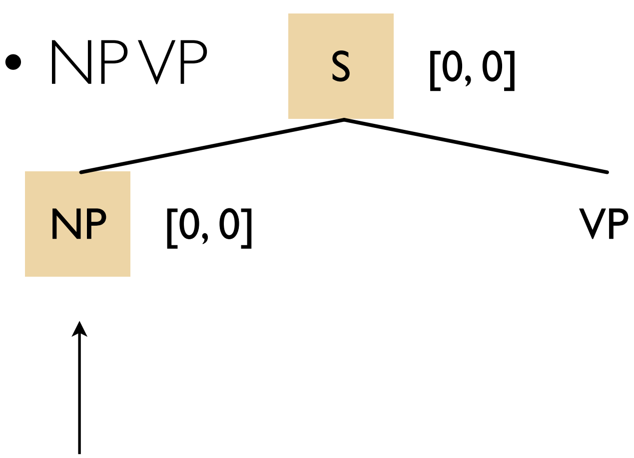


Predict the rule S → • NP VP

Example run

0 I 1 prefer 2 a 3 morning 4 flight 5

S → • NP VP



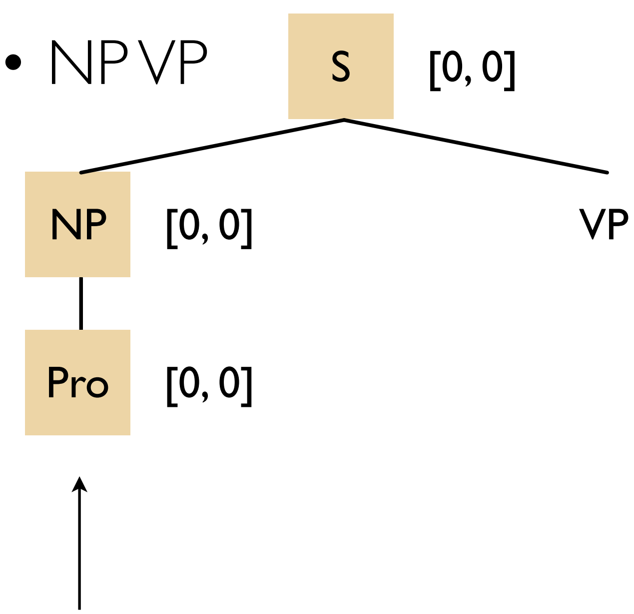
Predict the rule NP → • Pro

Example run

0 I 1 prefer 2 a 3 morning 4 flight 5

S → • NP VP

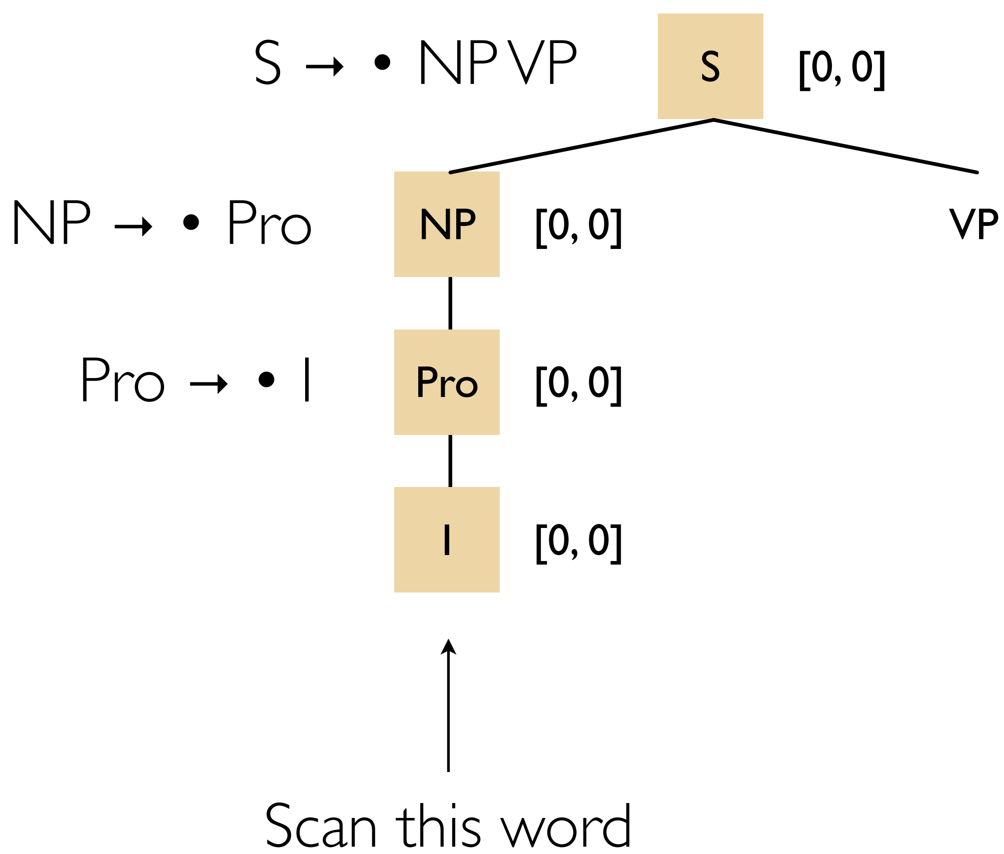
NP → • Pro



Predict the rule Pro → • I

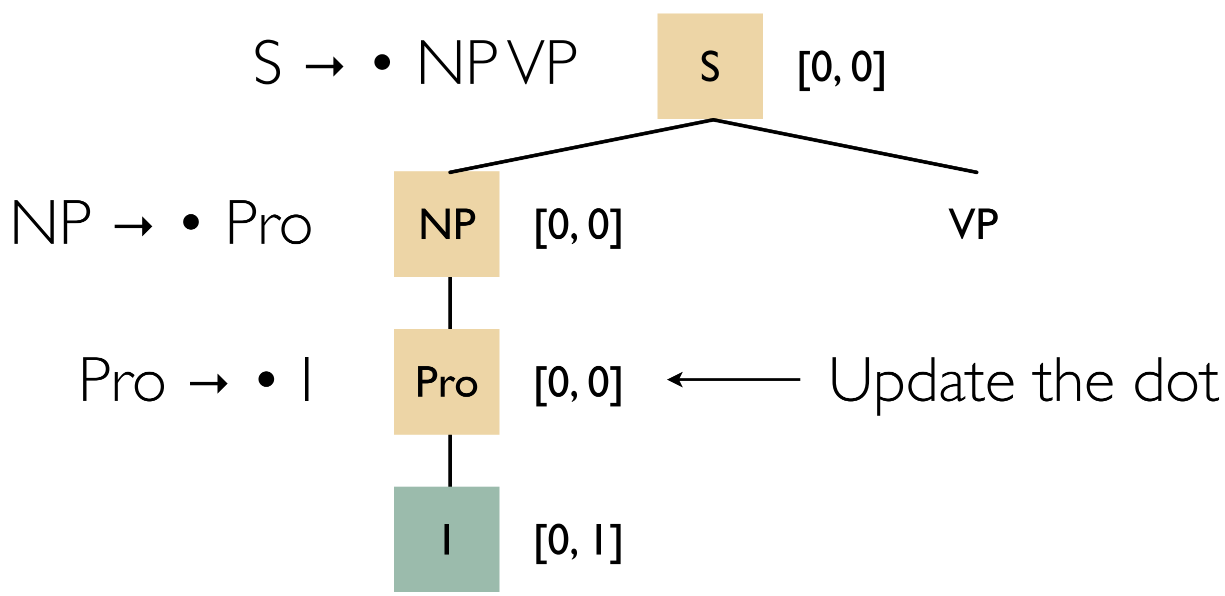
Example run

0 I 1 prefer 2 a 3 morning 4 flight 5



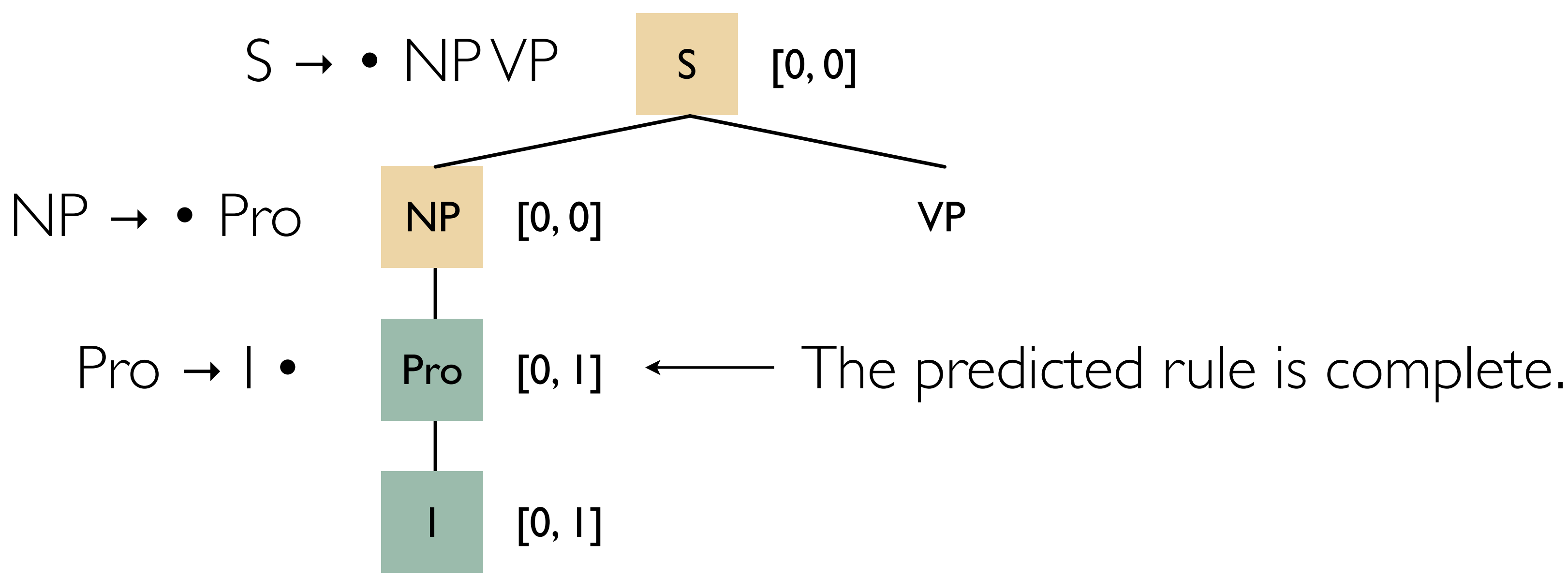
Example run

0 I 1 prefer 2 a 3 morning 4 flight 5



Example run

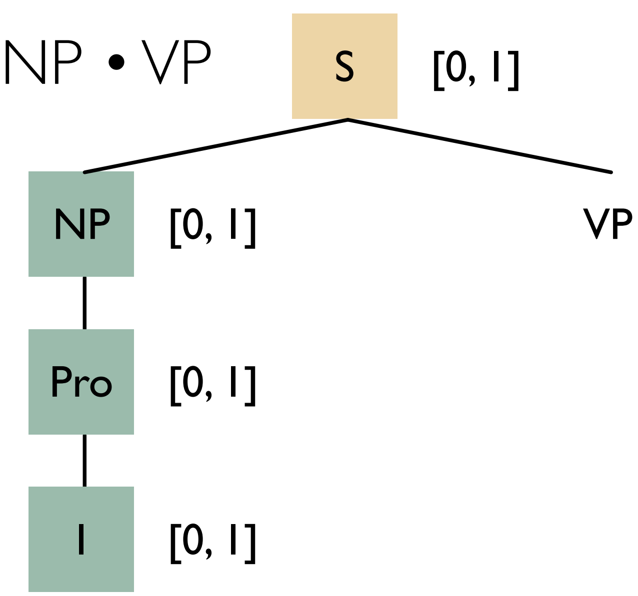
0 I 1 prefer 2 a 3 morning 4 flight 5



Example run

0 I 1 prefer 2 a 3 morning 4 flight 5

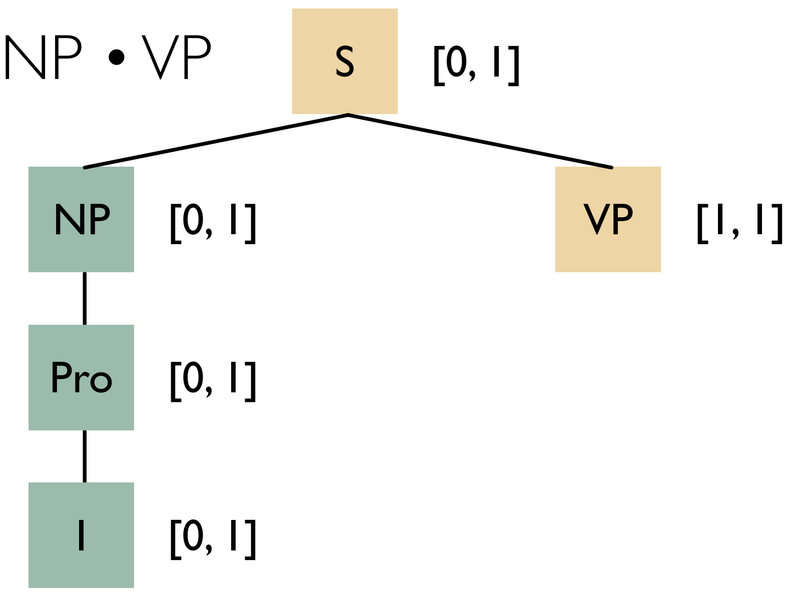
S → NP • VP



Example run

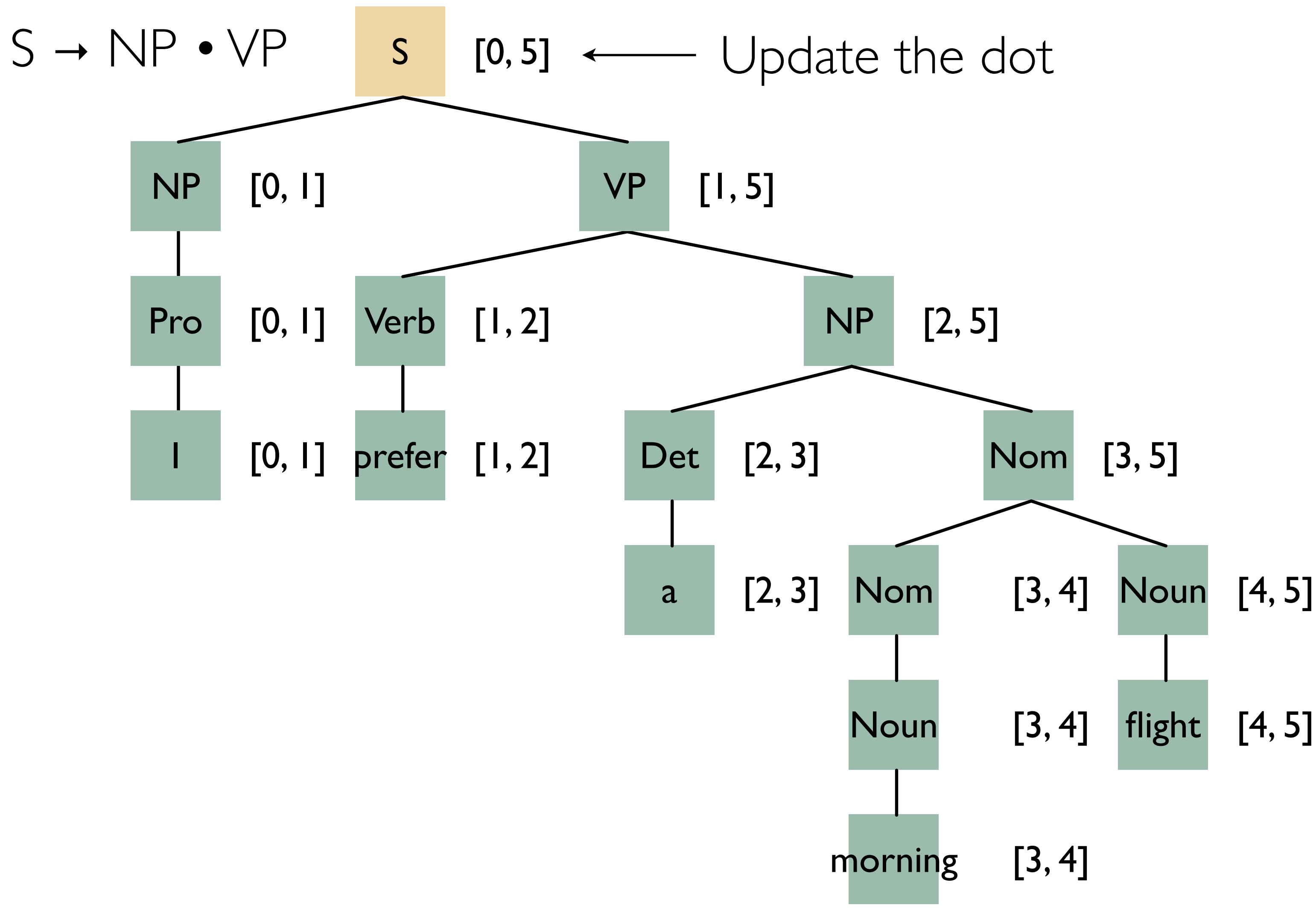
0 I 1 prefer 2 a 3 morning 4 flight 5

S → NP • VP



Example run

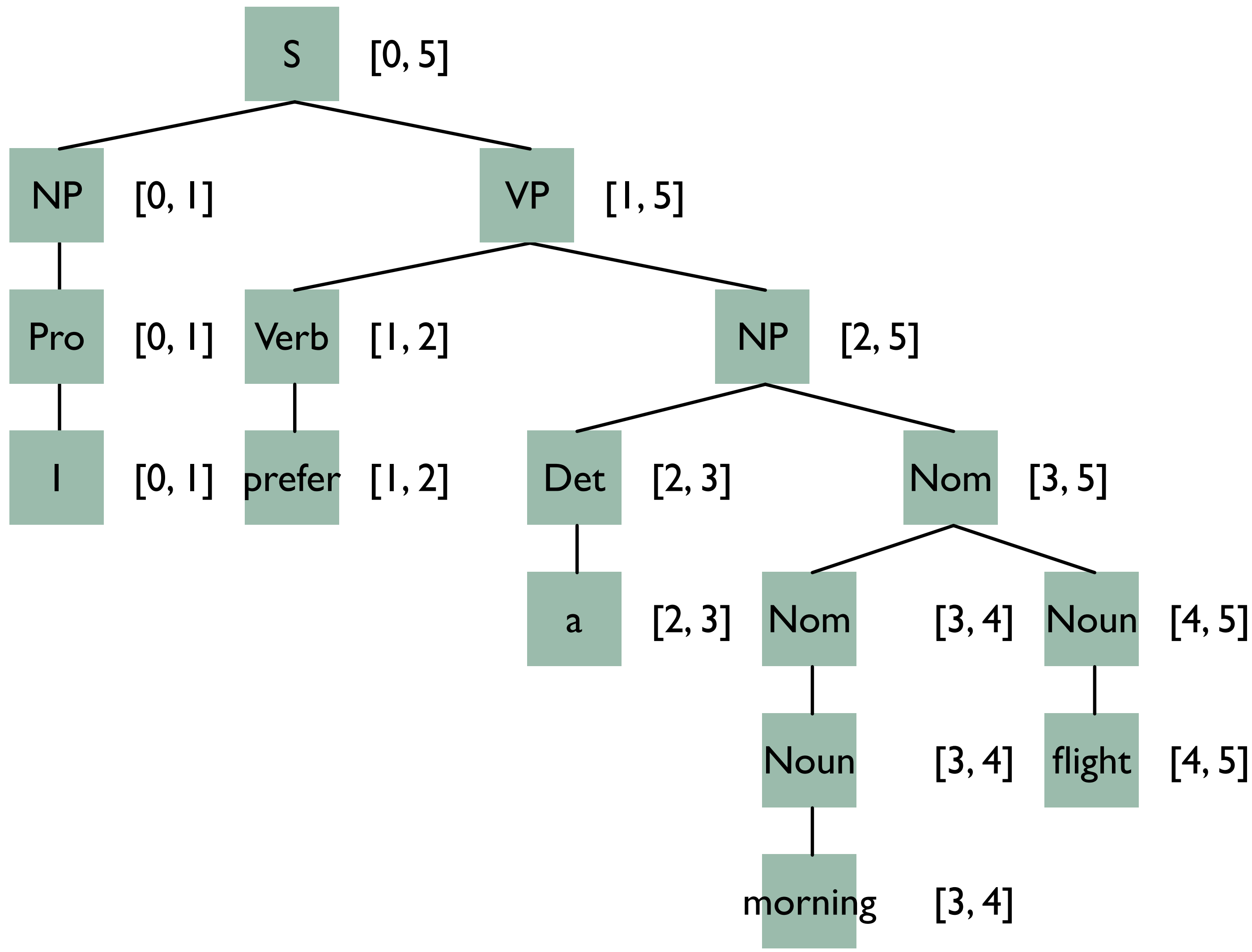
0 I 1 prefer 2 a 3 morning 4 flight 5



The Earley algorithm

Example run

0 I 1 prefer 2 a 3 morning 4 flight 5



The algorithm

- Start with the start symbol *S*.
- Take the leftmost nonterminal and predict all possible expansions.
- If the next symbol in the expansion is a word, match it against the input sentence (scan); otherwise, repeat.
- If there is nothing more to expand, the subtree is complete; in this case, continue with the next incomplete subtree.

Dotted rules

- A dotted rule is a partially processed rule.

  Example: S → NP • VP

- The dot can be placed in front of the first symbol, behind the last symbol, or between two symbols on the right-hand side of a rule.

- The general form of a dotted rule thus is $A \rightarrow \alpha \bullet \beta$, where $A \rightarrow \alpha\beta$ is the original, non-dotted rule.

Chart entries

- The chart contains entries of the form [min, max, A → α • β], where min and max are positions in the input and A → α • β is a dotted rule.
- Such an entry says: 'We have built a parse tree whose first rule is A → αβ and where the part of this rule that corresponds to α covers the words between min and max.'

Inference rules

Axiom

$$[0, 0, S \to \bullet\ \alpha]$$

$S \to \alpha$

Predict

$$\frac{[i, j, A \to \alpha \bullet B\ \beta]}{[j, j, B \to \bullet\ \gamma]}$$

$B \to \gamma$

Scan

$$\frac{[i, j, A \to \alpha \bullet a\ \beta]}{[i, j + 1, A \to \alpha\ a \bullet \beta]}$$

$w_j = a$

Complete

$$\frac{[i, j, A \to \alpha \bullet B\ \beta] \quad [j, k, B \to \gamma\ \bullet]}{[i, k, A \to \alpha\ B \bullet \beta]}$$

Pseudo code 1

```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE((γ → • S, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
   for each state in chart[i] do
     if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
         PREDICTOR(state)
     elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
         SCANNER(state)
     else
         COMPLETER(state)
   end
  end
  return(chart)
```

Pseudo code 2

```
procedure PREDICTOR((A → α • B β, [i, j]))
    for each (B → γ) in GRAMMAR-RULES-FOR(B, grammar) do
        ENQUEUE((B → • γ, [j, j]), chart[j])
    end

procedure SCANNER((A → α • B β, [i, j]))
    if B ⊂ PARTS-OF-SPEECH(word[j]) then
        ENQUEUE((B → word[j], [j, j+1]), chart[j+1])

procedure COMPLETER((B → γ •, [j, k]))
    for each (A → α • B β, [i, j]) in chart[j] do
        ENQUEUE((A → α B • β, [i, k]), chart[k])
    end
```

Recogniser/parser

- When parsing is complete, is there a chart entry? [0, n, S → α •]
  - Recognizer
- If we want a parser, we have to add back pointers, and retrieve a tree
- Earley's algorithm can be used for PCFGs, but it is more complicated than for CKY

Summary

- The Earley algorithm is a parsing algorithm for arbitrary context-free grammars.
- In contrast to the CKY algorithm, it also uses top–down information.
- Also in contrast to the CKY algorithm, its probabilistic extension is not straightforward.
- Reading: J&M 13.4.2