

Computerlinguistik I

Vorlesung im WiSe 2017/2018
(M-GSW-09)

Prof. Dr. Udo Hahn

Lehrstuhl für Computerlinguistik
Institut für Germanistische Sprachwissenschaft
Friedrich-Schiller-Universität Jena

<http://www.julielab.de>

der folgende Teil der Vorlesung
ist einer Ausarbeitung entnommen von

Dr. Christel Kemke

Department of Computer Science
University of Manitoba

Sample Grammar

Grammar (S, NT, T, P) – Sentence Symbol S \in NT, Part-of-Speech \subseteq NT, syntactic Constituents \subseteq NT, Grammar Rules P \subseteq NT \times (NT \cup T)*

S \rightarrow NP VP statement

S \rightarrow Aux NP VP question

S \rightarrow VP command

NP \rightarrow Det Nominal

NP \rightarrow Proper-Noun

Nominal \rightarrow Noun | Noun Nominal | Nominal PP

VP \rightarrow Verb | Verb NP | Verb PP | Verb NP PP

PP \rightarrow Prep NP

Det \rightarrow that | this | a

Noun \rightarrow book | flight | meal | money

Proper-Noun \rightarrow Houston | American Airlines | TWA

Verb \rightarrow book | include | prefer

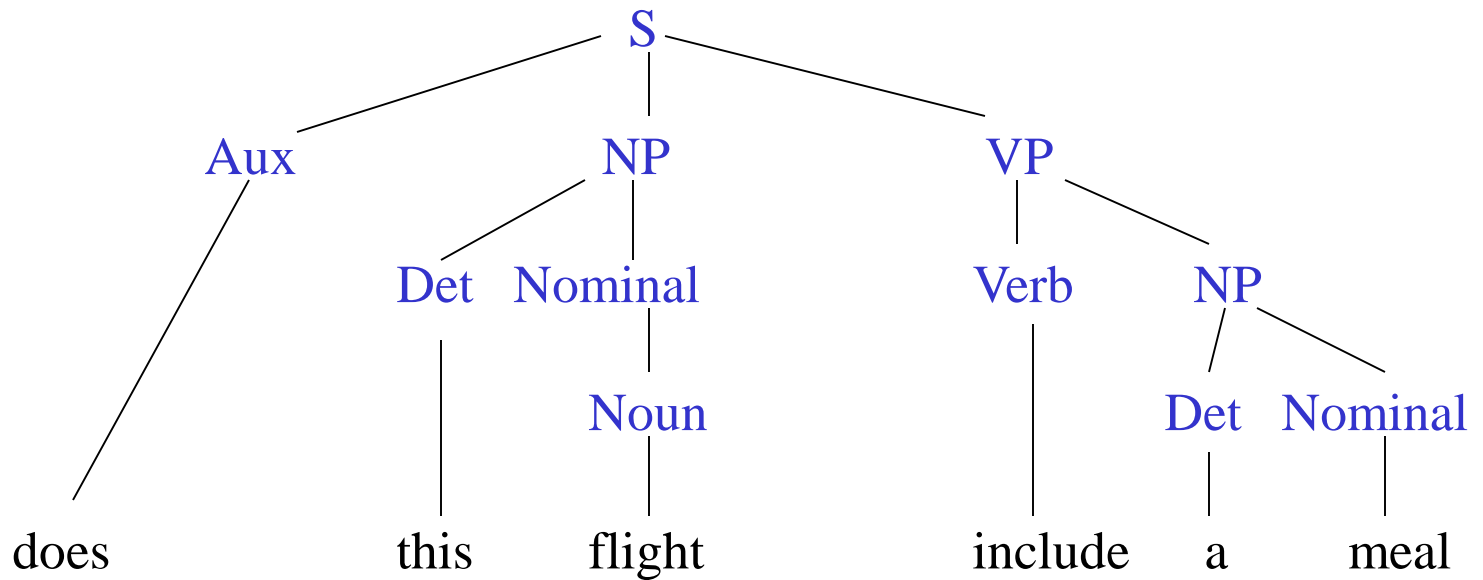
Aux \rightarrow does

Prep \rightarrow from | to | on

Task: Parse "Does this flight include a meal?"

Sample Parse Tree

Task: Parse "Does this flight include a meal?"



Problems with Bottom-up and Top-down Parsing

Problems with **left-recursive rules** like $NP \rightarrow NP PP$:
don't know how many times recursion is needed (top-down)

Pure Bottom-up or Top-down Parsing is **inefficient** because it generates and explores too many structures which in the end turn out to be invalid (several grammar rules applicable \rightarrow 'interim' ambiguity).

Combine top-down and bottom-up approach:

Start with sentence; use rules top-down (**look-ahead**);
read input; try to find shortest path from input to highest unparsed constituent (from **left to right**).

\rightarrow **Chart-Parsing / Earley-Parser**

Chart Parsing / Earley Algorithm

Earley-Parser based on Chart-Parsing

Essence: Integrate top-down and bottom-up parsing. Keep recognized sub-structures (sub-trees) for shared use during parsing.

Top-down: Start with S-symbol. Generate all applicable rules for S. Go further down with left-most constituent in rules and add rules for these constituents until you encounter a left-most node on the RHS which is a word category (POS).

Bottom-up: Read input word and compare. If word matches, mark as recognized and move parsing on to the next category in the rule(s).

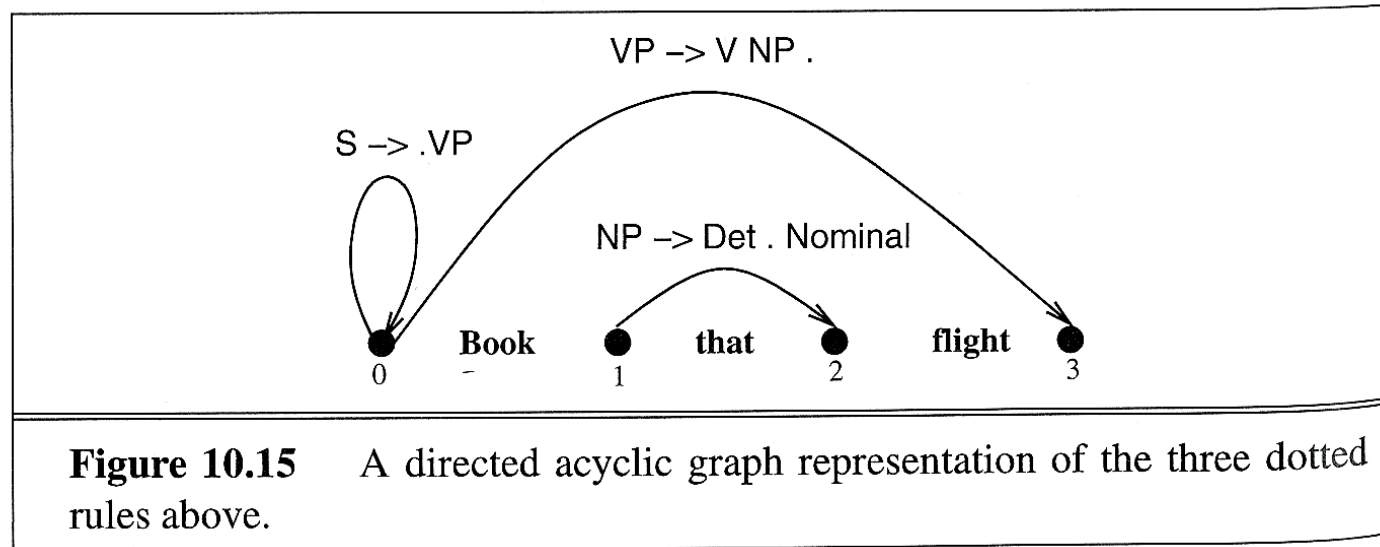
Chart – a Graph-Based Data Structure for Parsing

Chart

Sequence of n input words; $n+1$ nodes marked 0 to n .

Arcs indicate recognized part of RHS of rule.

The • indicates recognized constituents in rules.



Jurafsky & Martin, Figure 10.15, p. 380

Chart Parsing / Earley Parser 1

Chart

Sequence of input words; $n+1$ nodes marked 0 to n .

States in chart represent possible rules and recognized constituents, with arcs.

Interim state

$S \rightarrow \bullet VP, [0,0]$

- top-down look at rule $S \rightarrow VP$
- nothing of RHS of rule yet recognized (\bullet is far left)
- arc at beginning, no coverage (covers no input word; beginning of arc at 0 and end of arc at 0)

Chart Parsing / Earley Parser 2

Interim states

NP → Det • Nominal, [1,2]

- top-down look with rule NP → Det • Nominal
- Det recognized (• after Det)
- arc covers one input word which is between node 1 and node 2
- look next for Nominal

NP → Det Nominal •, [1,3]

- Nominal was recognized, move • after Nominal
- move end of arc to cover Nominal (change 2 to 3)
- structure is completely recognized; arc is inactive; mark NP as recognized in other rules (move •).

Chart - 0

$S \rightarrow \cdot VP$

$VP \rightarrow \cdot V NP$

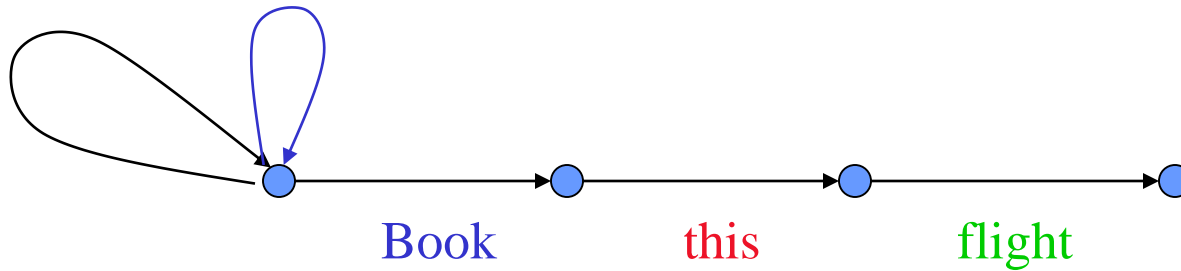


Chart - 1

$S \rightarrow \cdot VP$

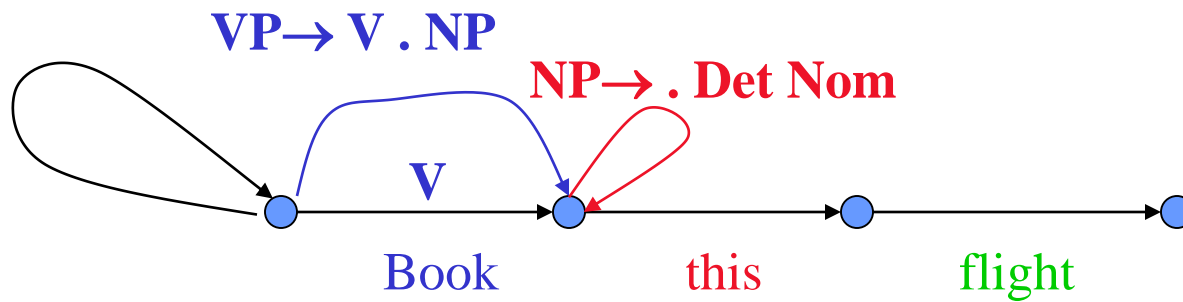


Chart - 2

$S \rightarrow \cdot VP$

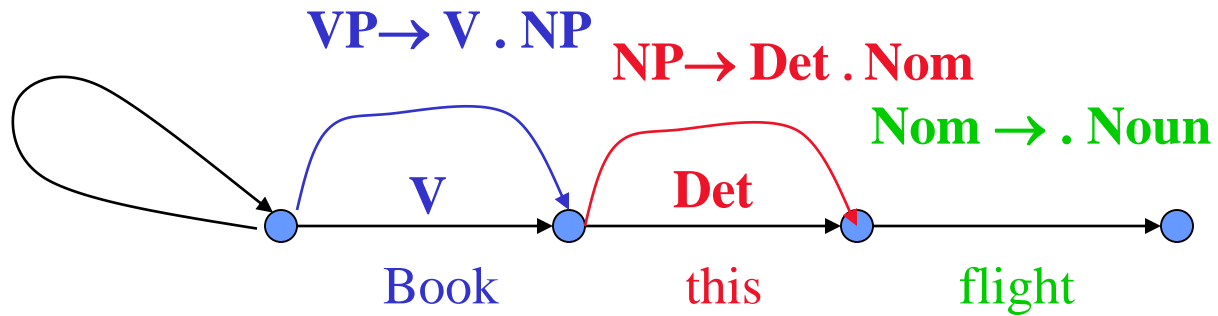


Chart - 3a

$S \rightarrow \cdot VP$

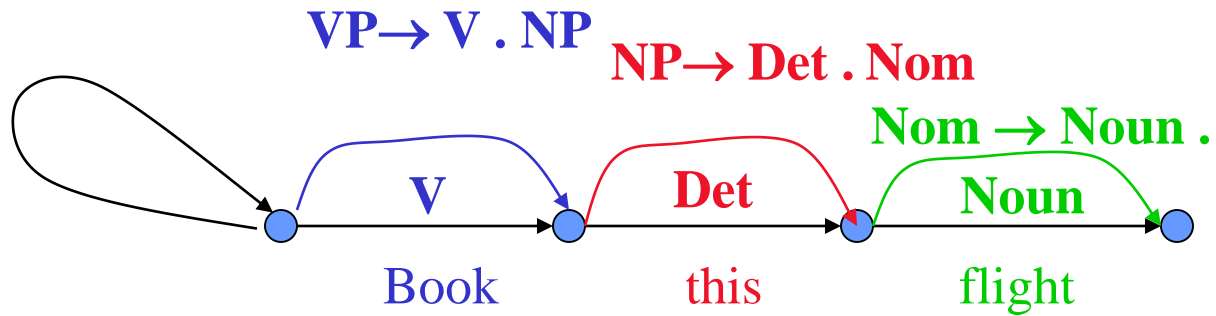


Chart - 3b

$S \rightarrow \cdot VP$

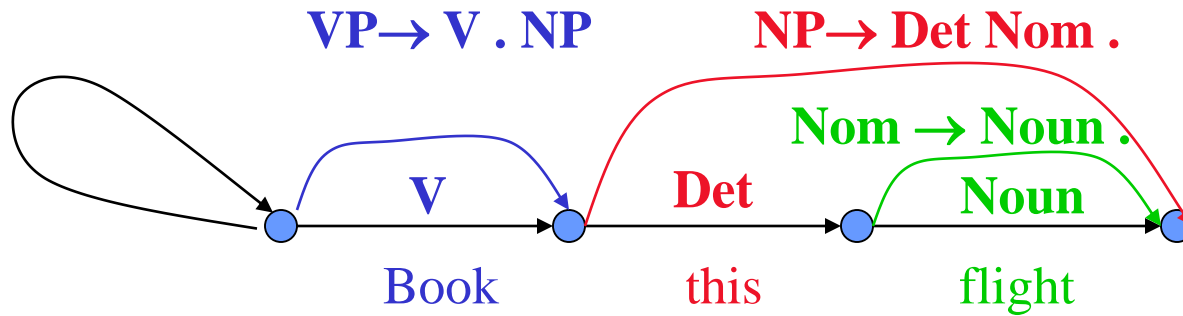


Chart - 3c

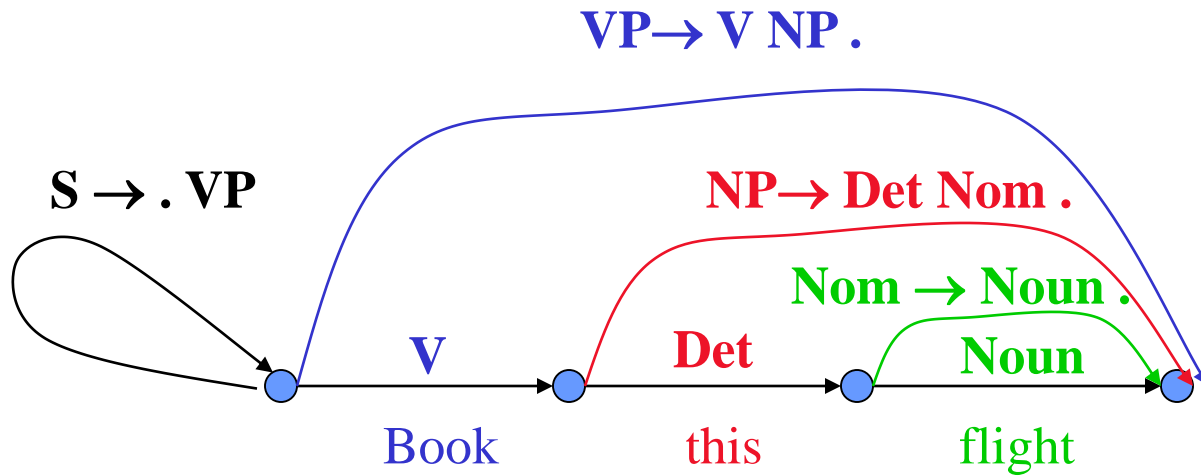


Chart - 3d

$S \rightarrow VP.$

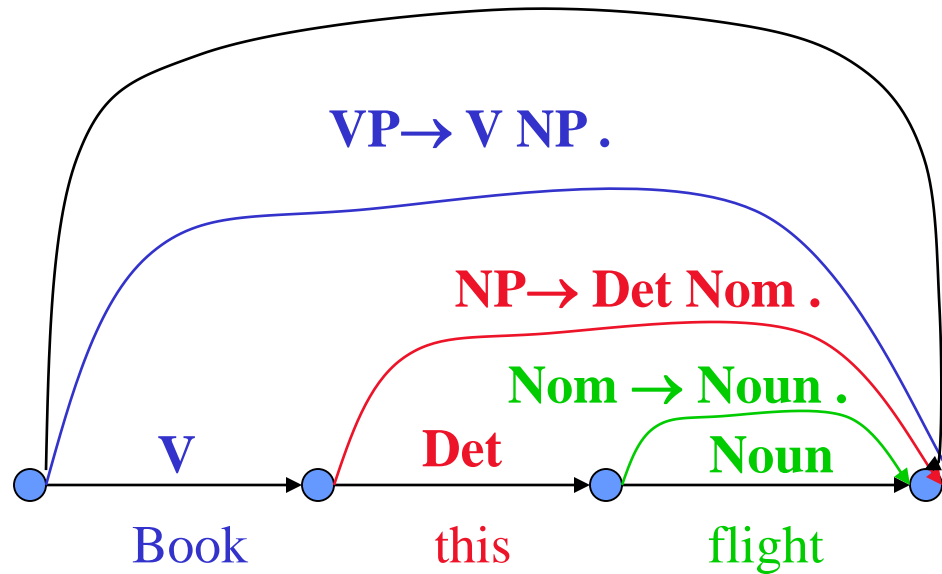


Chart - All States

$S \rightarrow VP.$

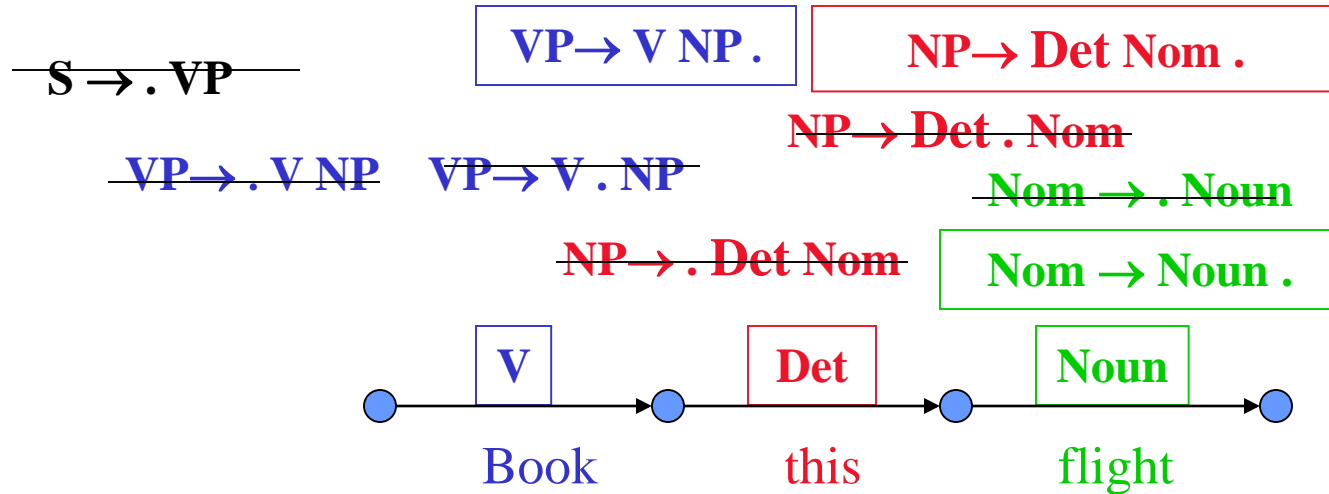


Chart - Final States

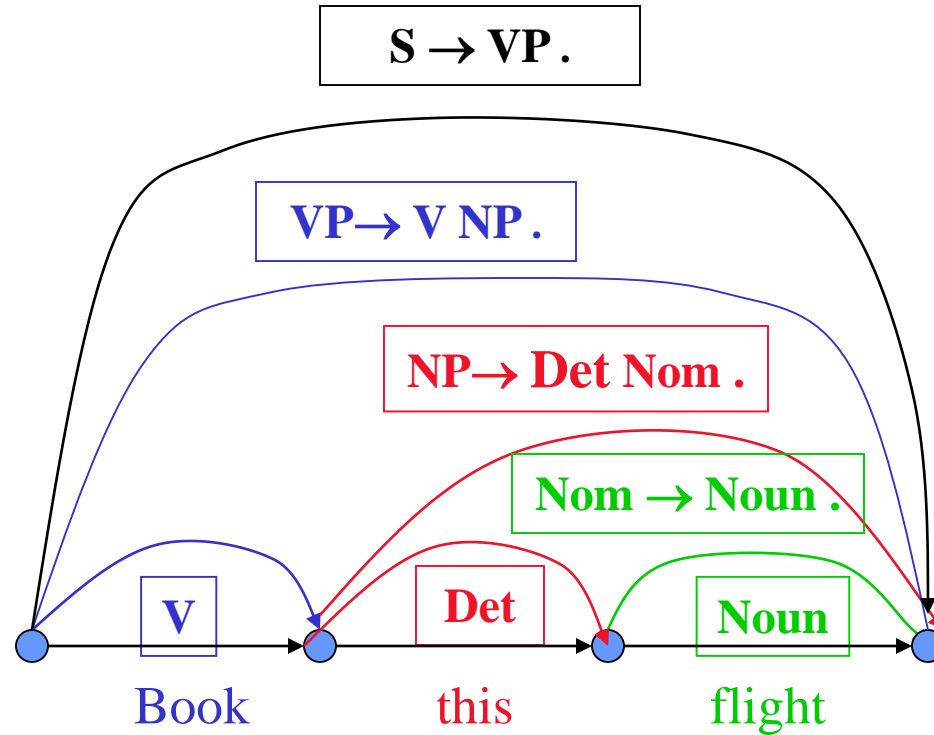


Chart 0 with two S- and two VP-Rules

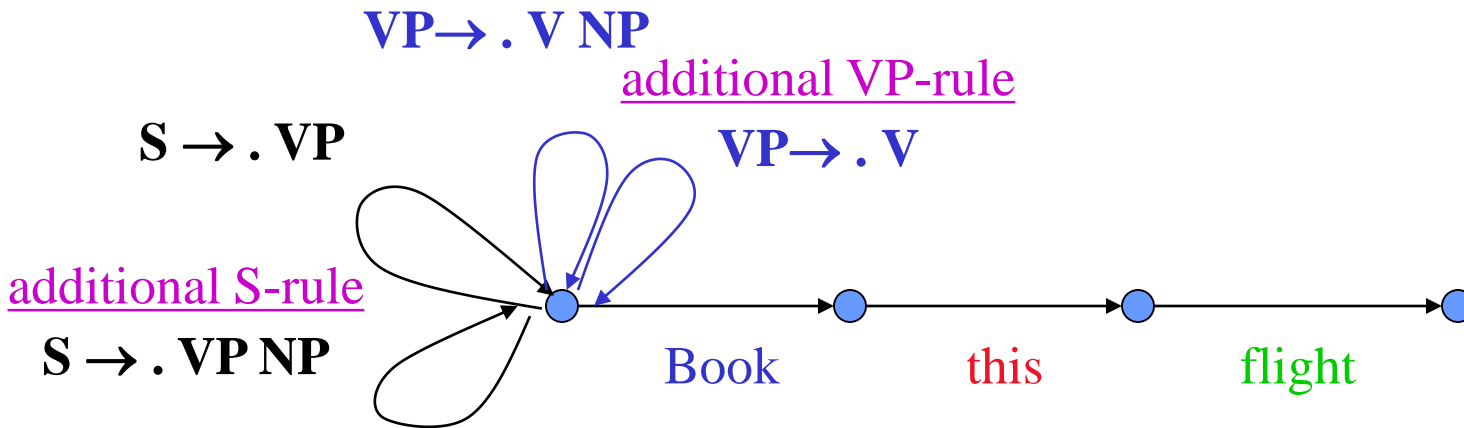


Chart 1a with two S- and two VP-Rules

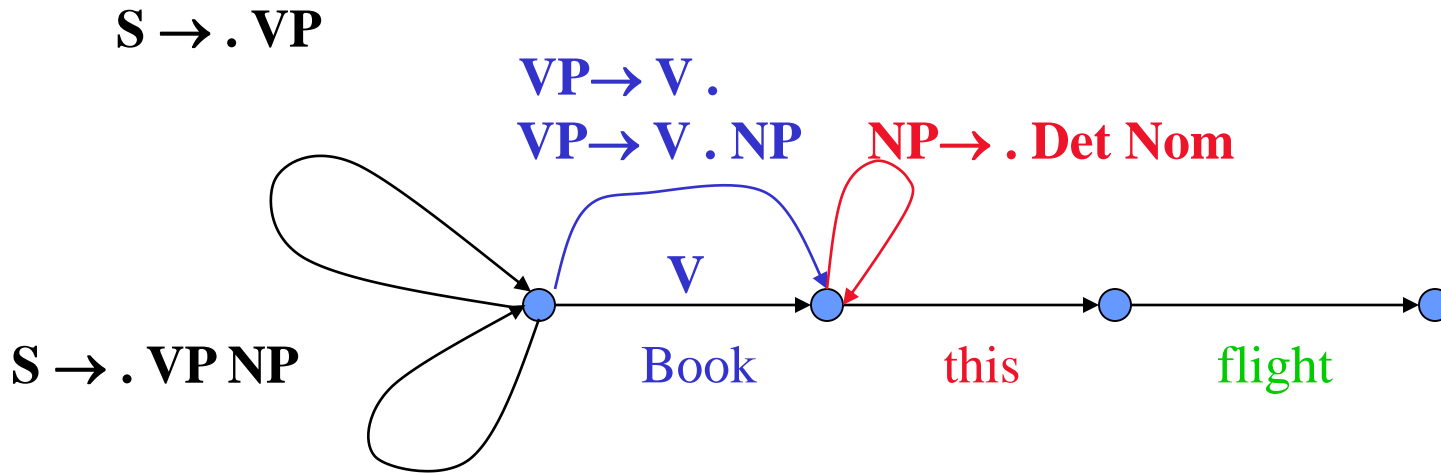


Chart 1b with two S- and two VP-Rules

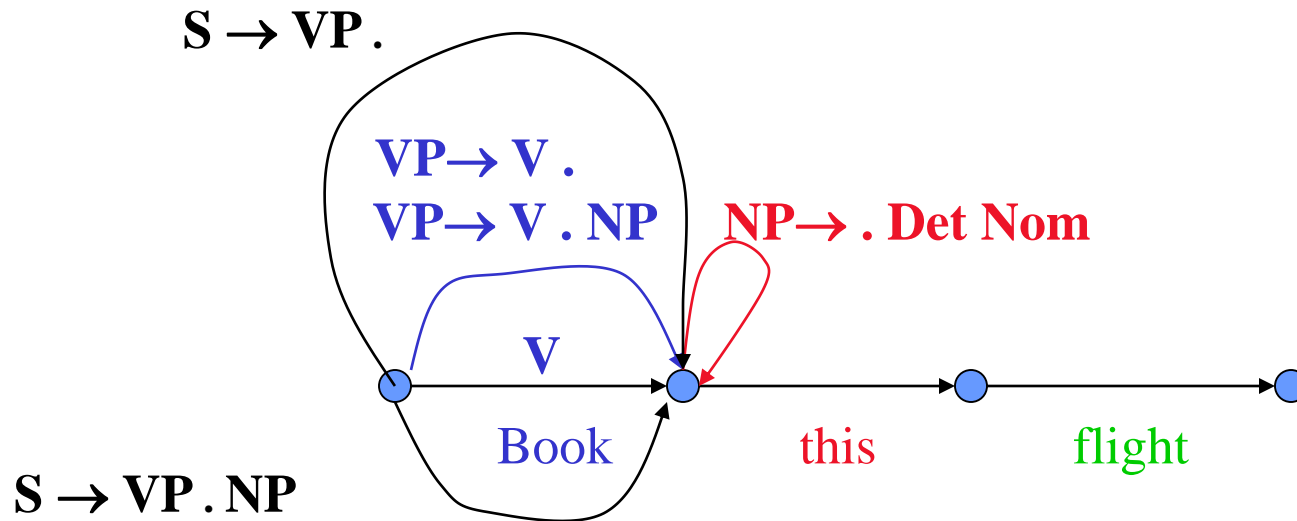


Chart 2 with two S- and two VP-Rules

$S \rightarrow VP.$

$S \rightarrow VP.NP$

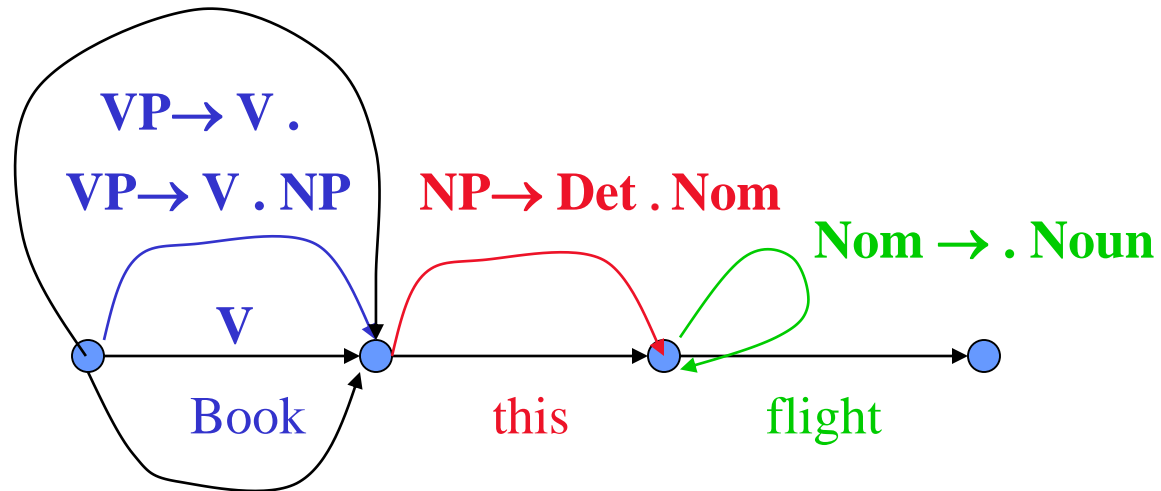
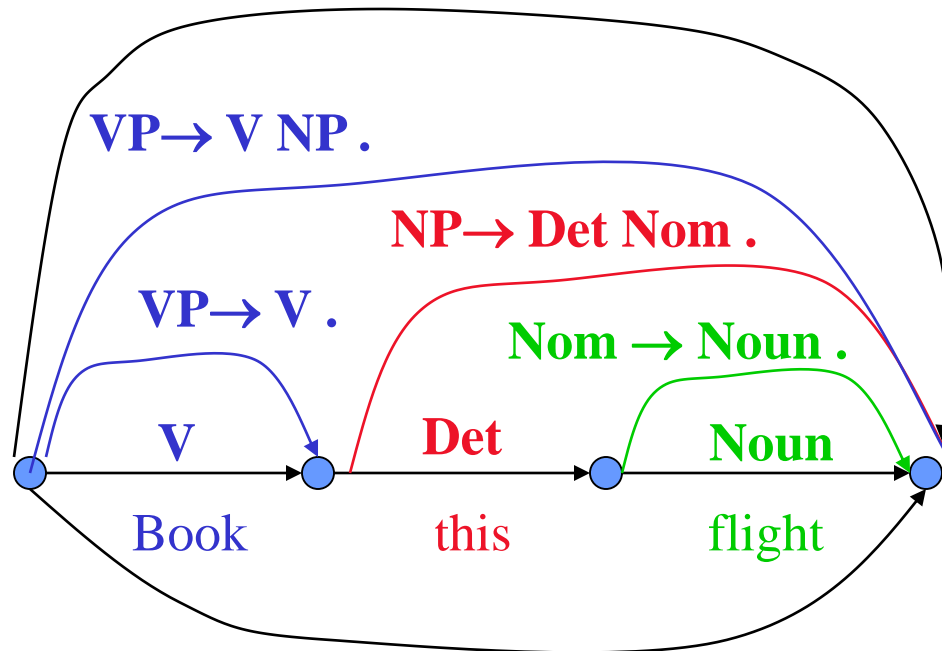


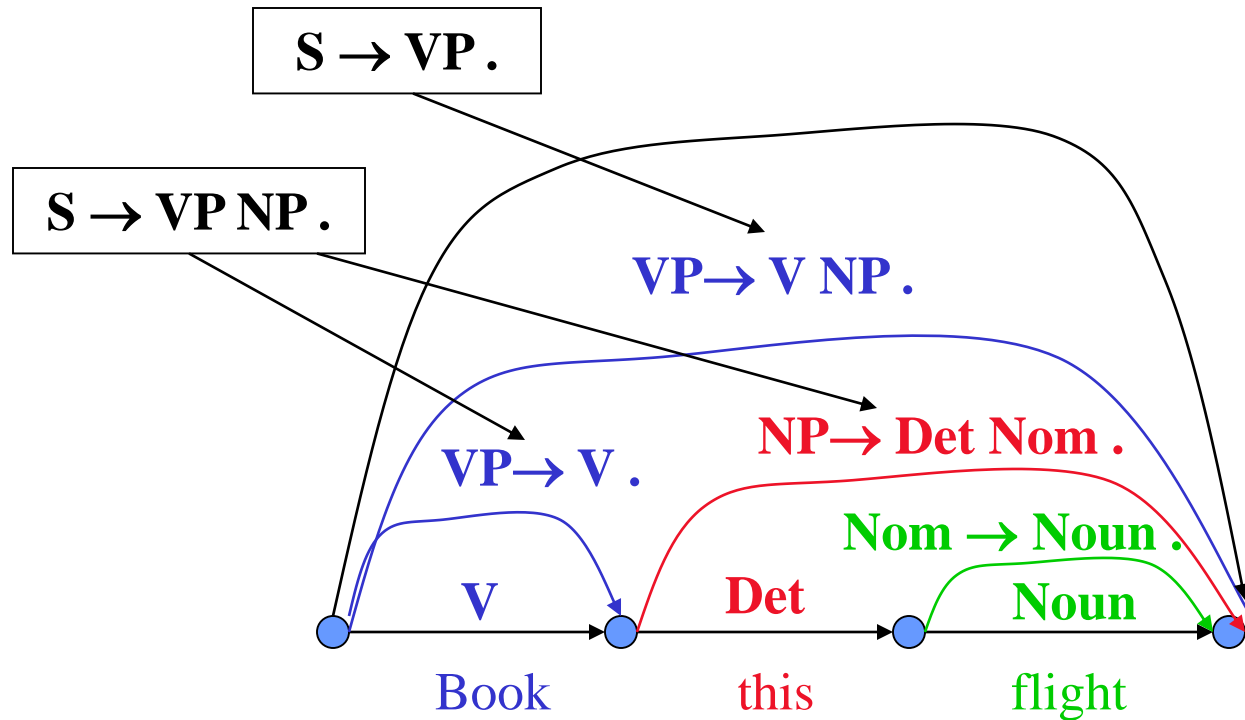
Chart 3 with two S- and two VP-Rules

$S \rightarrow VP.$



$S \rightarrow VP NP.$

Final Chart - with two S-and two VP-Rules



Earley Algorithm - Functions

predictor

generates new rules for partly recognized RHS with constituent right of • (**top-down generation**)

scanner

if word category (POS) is found right of the •, the Scanner reads the next input word and adds a rule for it to the chart (**bottom-up mode**)

completer

if rule is completely recognized (the • is far right), the recognition state of earlier rules in the chart advances: the • is moved over the recognized constituent (**bottom-up recognition**).

Earley-Algorithm

```
function EARLEY-PARSE(words, grammar) returns chart  
  ENQUEUE( $(\gamma \rightarrow \bullet S, [0,0])$ , 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)  
      else if INCOMPLETE?(state) and  
        NEXT-CAT(state) is a part of speech  
      then SCANNER(state)  
      else COMPLETER(state)  
    end  
  end  
  return(chart)
```

- continued on next slide -

```
procedure PREDICTOR(( $A \rightarrow \alpha \bullet B \beta$ ,  $[i, j]$ ))  
  for each ( $B \rightarrow \gamma$ ) in GRAMMAR-RULES-FOR( $B$ , grammar)  
  do ENQUEUE(( $B \rightarrow \gamma$   $[j, j]$ , chart[ $j$ ])  
end
```

```
procedure SCANNER (( $A \rightarrow \alpha \bullet B \beta$ ,  $[i, j]$ ))  
  if  $B \in$  PARTS-OF-SPEECH(word[ $j$ ])  
  then ENQUEUE(( $B \rightarrow$  word[ $j$ ],  $[j, j+1]$ ), chart[ $j+1$ ])
```

```
procedure COMPLETER (( $B \rightarrow \gamma \bullet$ ,  $[j, k]$ ))  
  for each ( $A \rightarrow \alpha \bullet B \beta$ ,  $[i, j]$ ) in chart[ $j$ ]  
  do ENQUEUE(( $A \rightarrow \alpha B \bullet \beta$ ,  $[i, k]$ ), chart[ $k$ ])  
end
```

```
procedure ENQUEUE(state, chart-entry)  
  if state is not already in chart-entry  
  then PUSH(state, chart-entry)
```

```
end
```

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

ENQUEUE($(\gamma \rightarrow \bullet S, [0, 0])$, *chart*[0])

for $i \leftarrow$ **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*)

procedure PREDICTOR($(A \rightarrow \alpha \bullet B \beta, [i, j])$)

for each $(B \rightarrow \gamma)$ **in** GRAMMAR-RULES-FOR(B , *grammar*) **do**

 ENQUEUE($(B \rightarrow \bullet \gamma, [j, j])$, *chart*[j])

end

procedure SCANNER($(A \rightarrow \alpha \bullet B \beta, [i, j])$)

if $B \subset$ PARTS-OF-SPEECH(*word*[j]) **then**

 ENQUEUE($(B \rightarrow \text{word}[j], [j, j+1])$, *chart*[$j+1$])

procedure COMPLETER($(B \rightarrow \gamma \bullet, [j, k])$)

for each $(A \rightarrow \alpha \bullet B \beta, [i, j])$ **in** *chart*[j] **do**

 ENQUEUE($(A \rightarrow \alpha B \bullet \beta, [i, k])$, *chart*[k])

end

procedure ENQUEUE(*state*, *chart-entry*)

if *state* is not already in *chart-entry* **then**

 PUSH(*state*, *chart-entry*)

end