Computerlinguistik II / Sprachtechnologie

Vorlesung im SoS 2017
(M-GSW-10)

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

Problems with Bottom-up and Top-down Parsing

Problems with left-recursive rules like NP → 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 → '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).

→ 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.

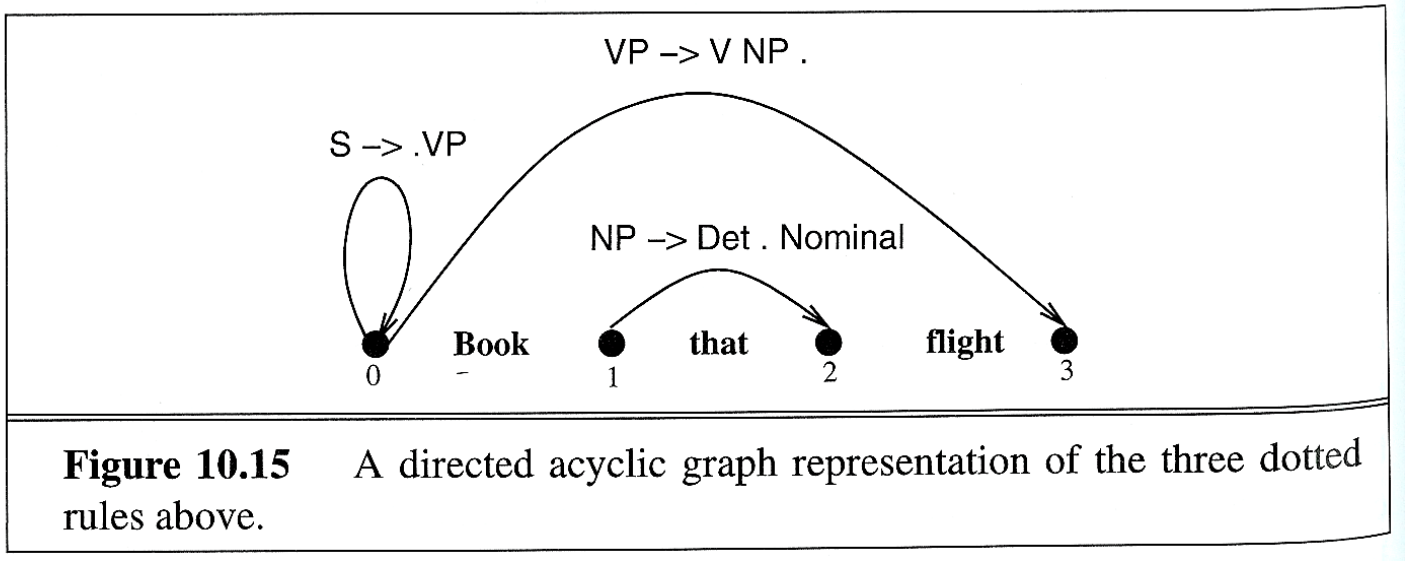


Figure 10.15 A directed acyclic graph representation of the three dotted rules above.

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 → • VP, [0,0]

- top-down look at rule S → VP
- nothing of RHS of rule yet recognized (• 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 → . VP

VP→ . V NP

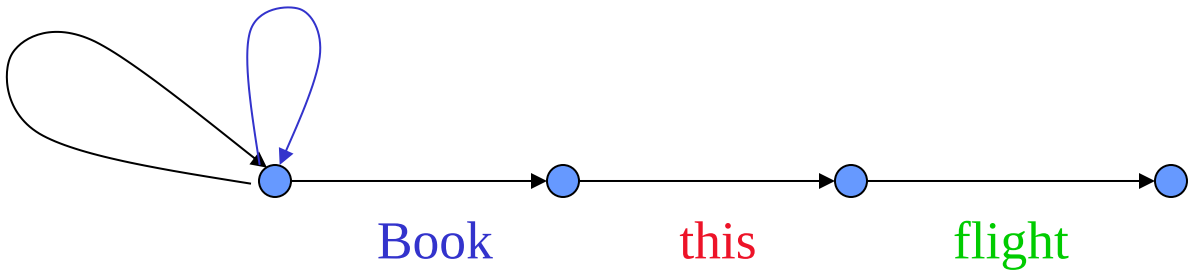


Chart - 1

S → . VP

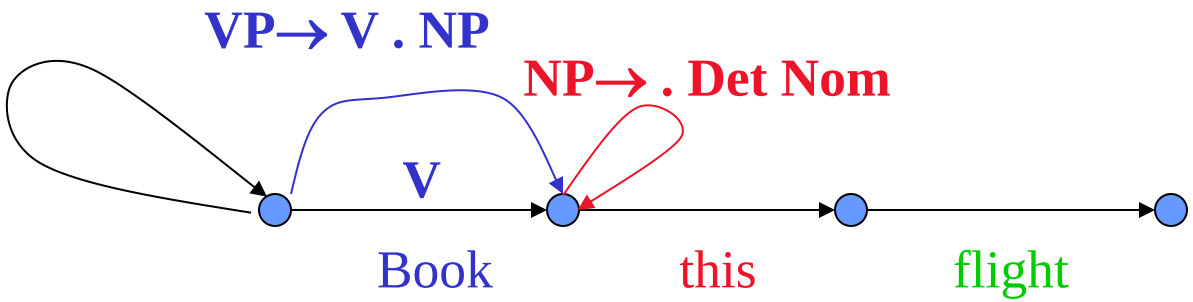


Chart - 2

S → . VP

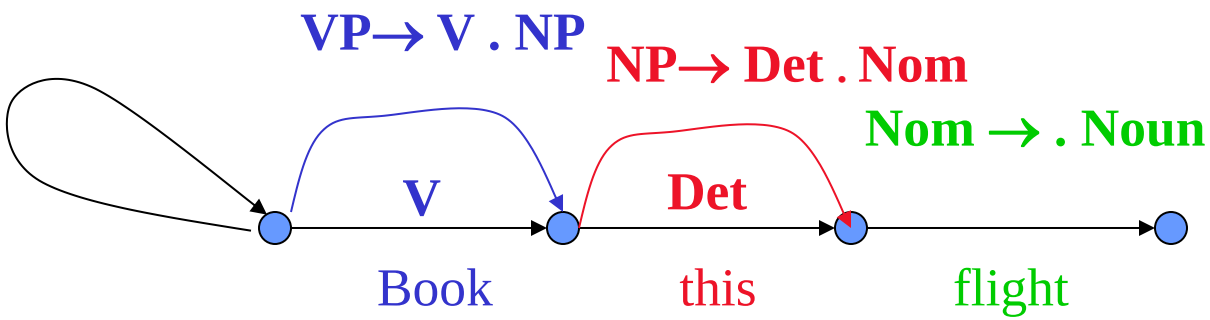


Chart - 3a

S → . VP

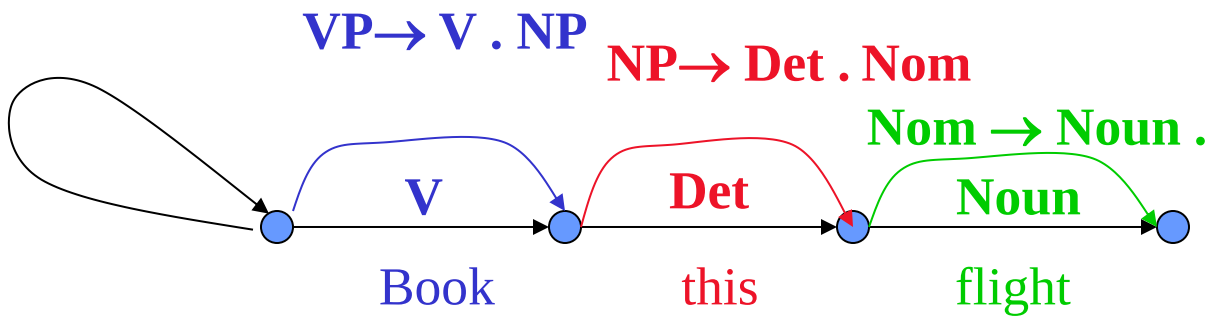


Chart - 3b

S → . VP

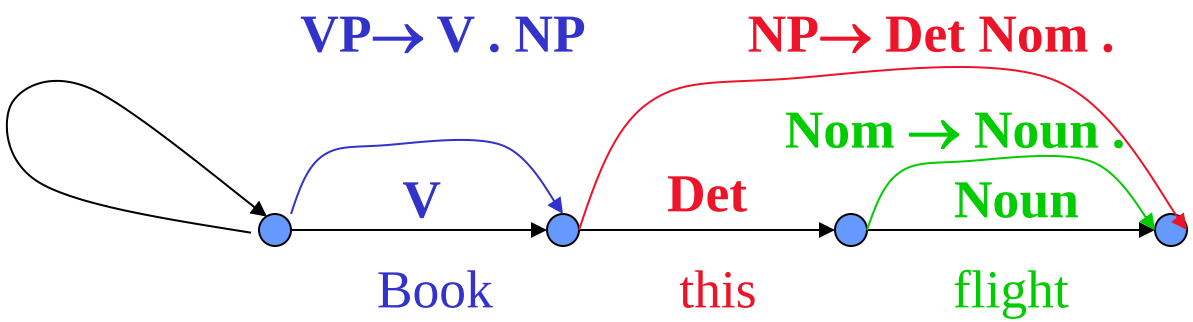


Chart - 3c

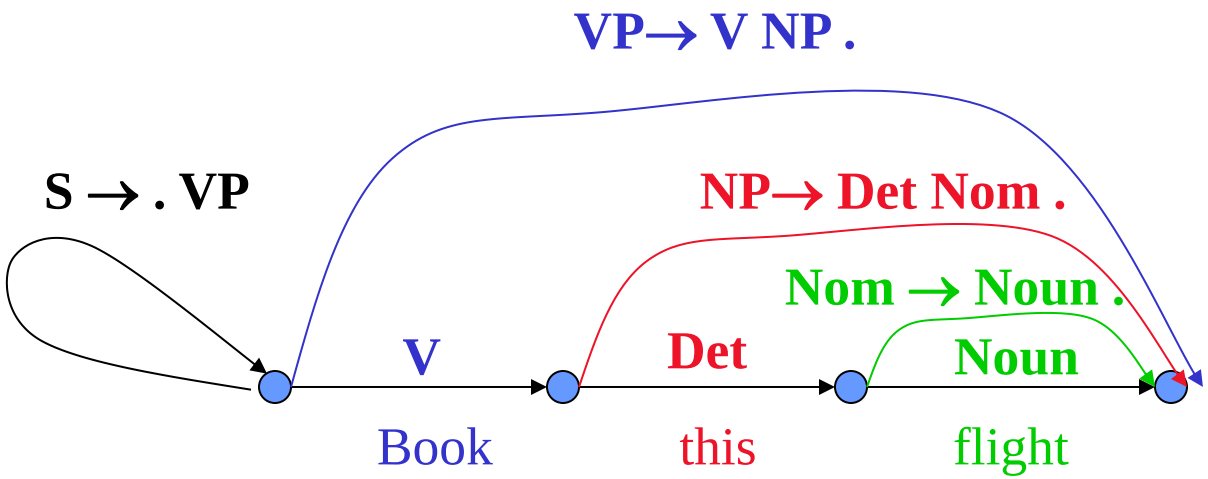


Chart - 3d

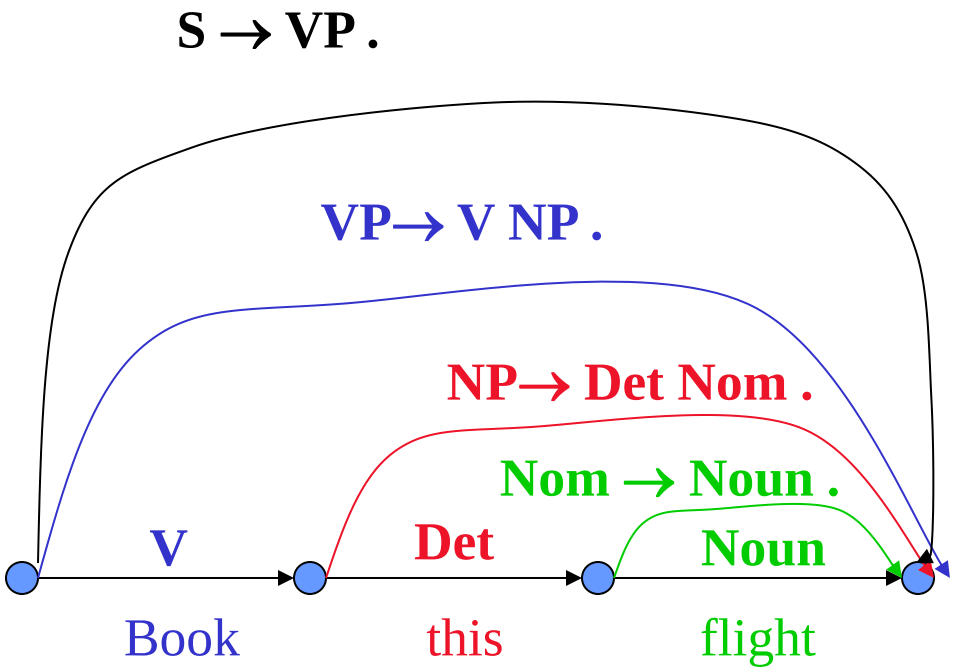


Chart - All States

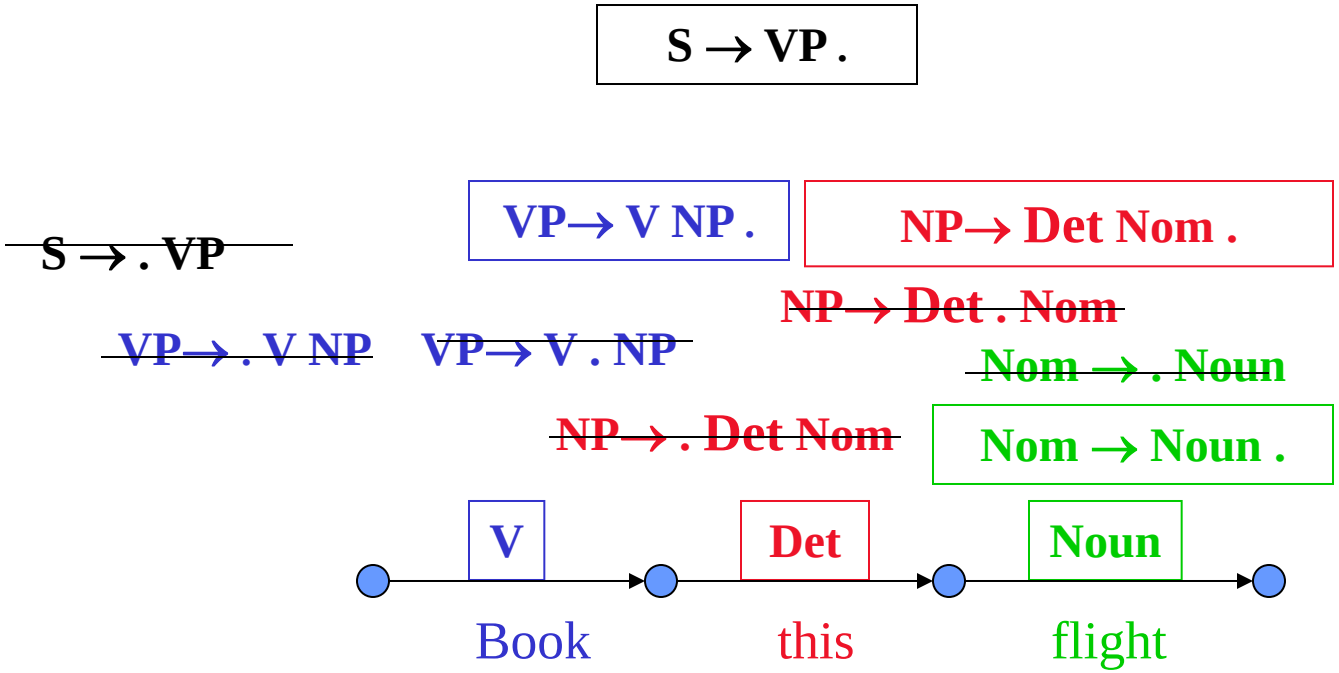


Chart - Final States

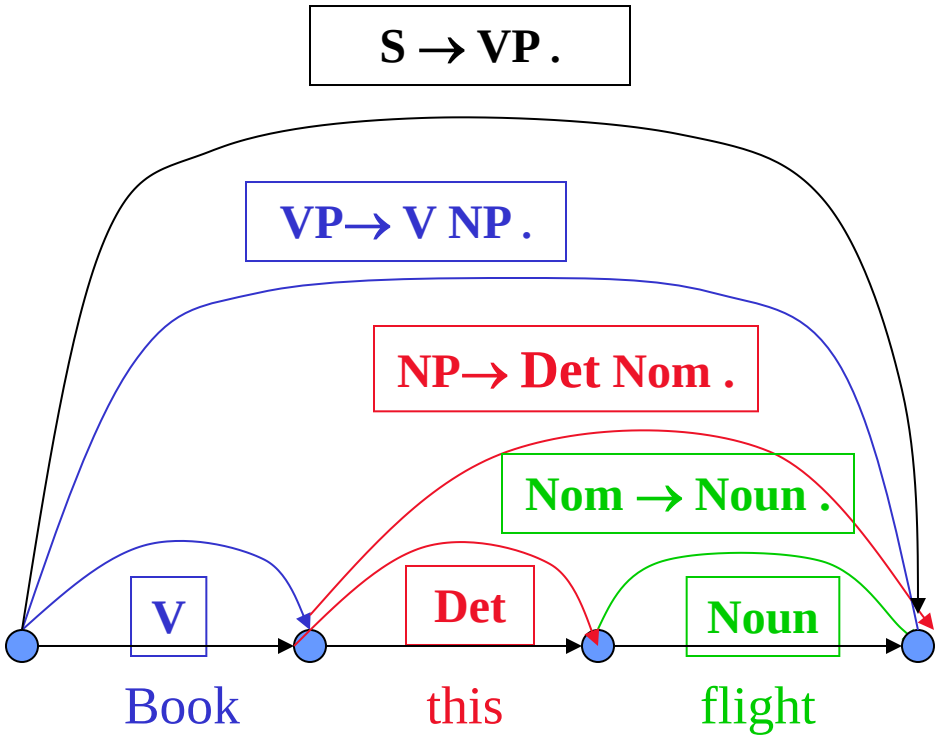


Chart 0 with two S- and two VP-Rules

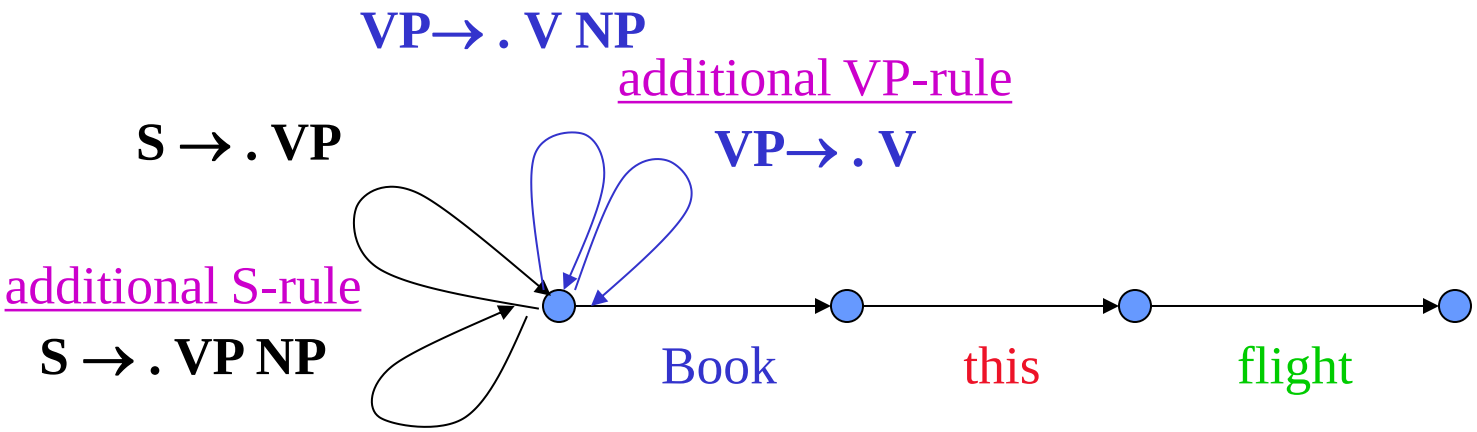


Chart 1a with two S- and two VP-Rules

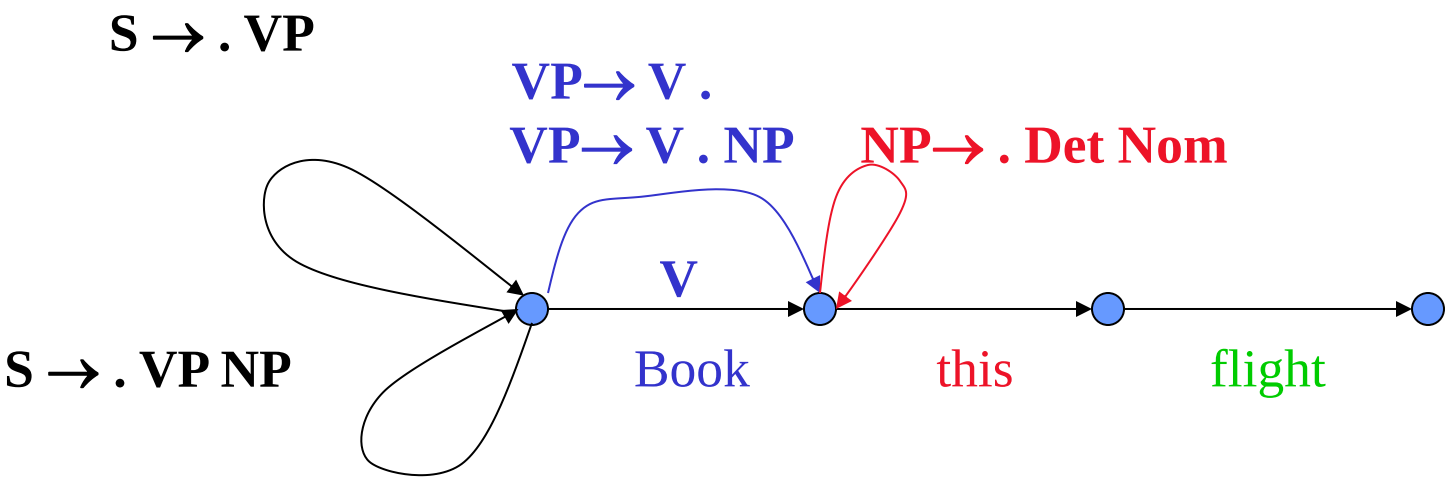


Chart 1b with two S- and two VP-Rules

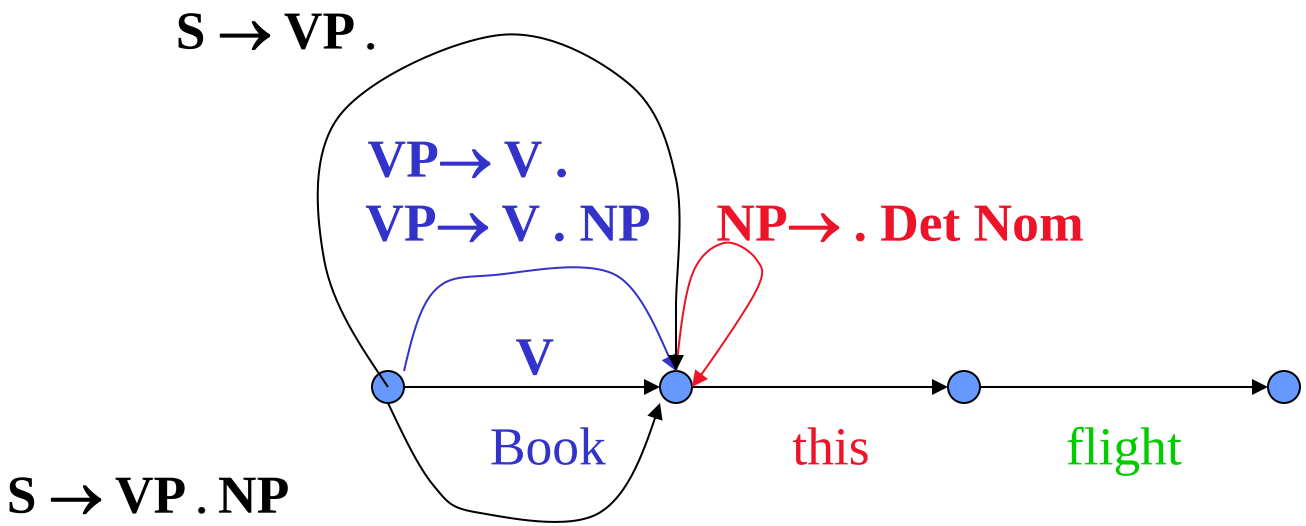


Chart 2 with two S- and two VP-Rules

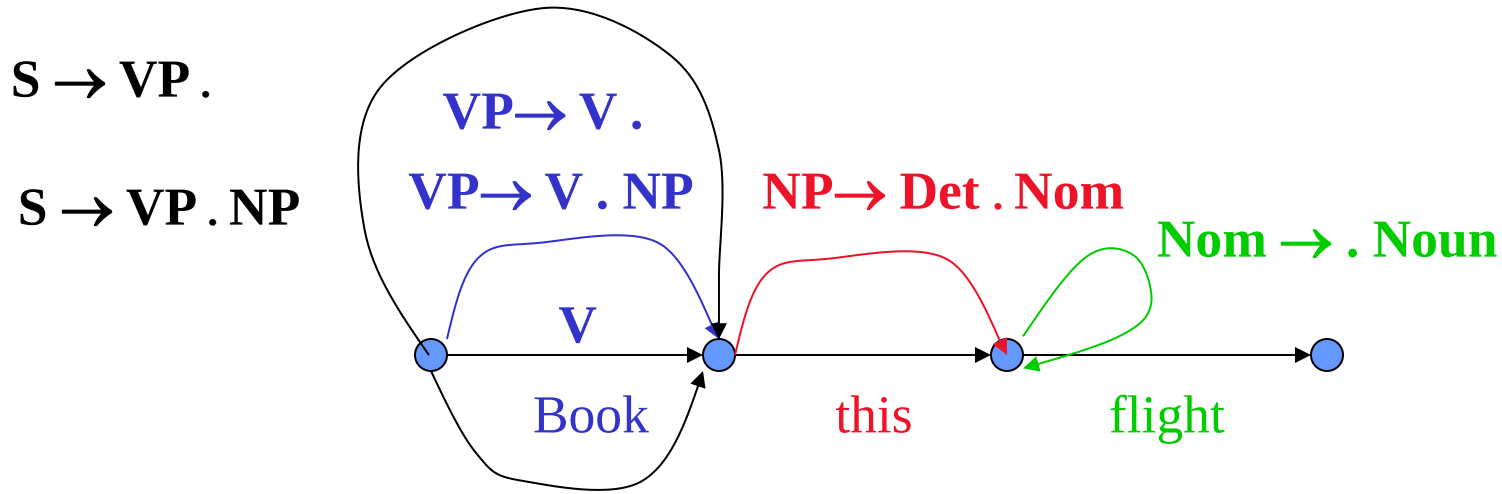
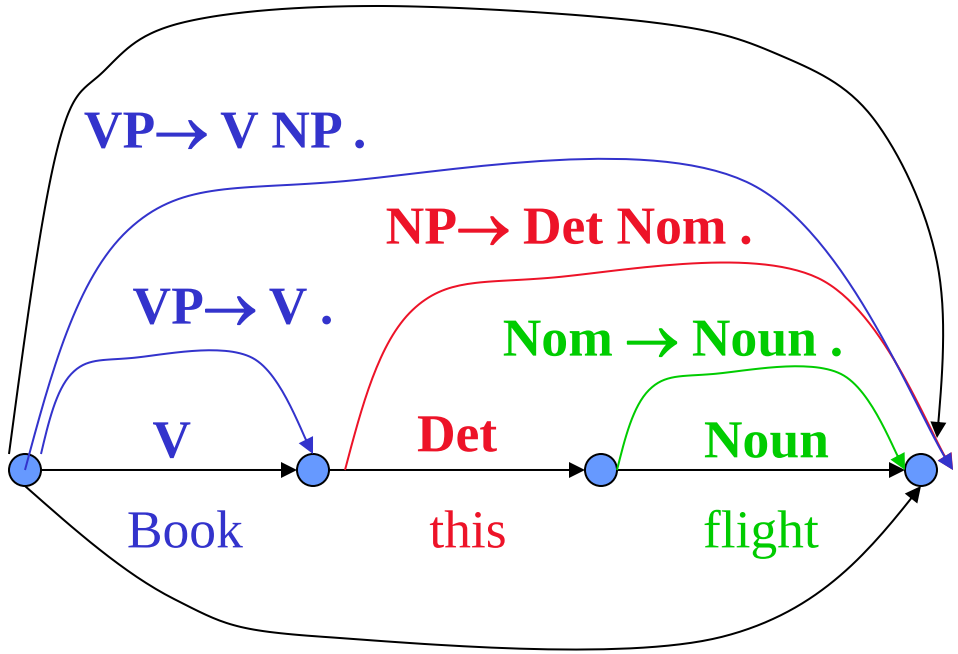


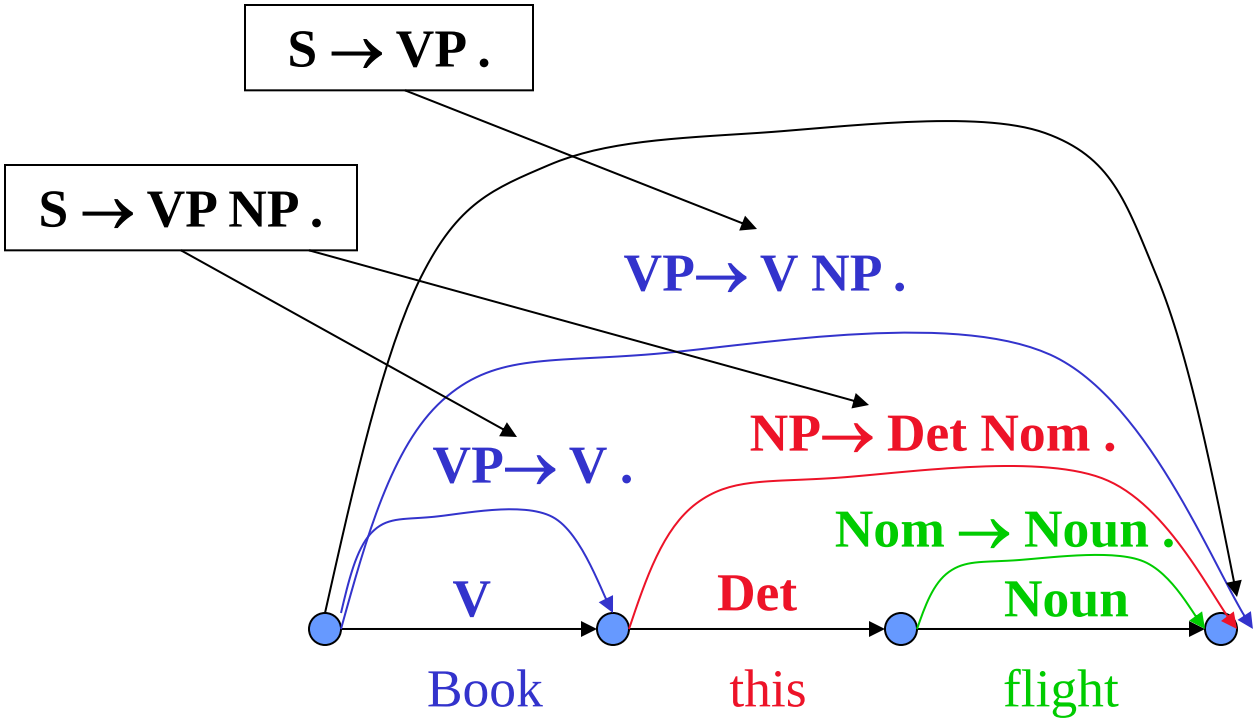
Chart 3 with two S- and two VP-Rules

S → VP .



S → 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((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*)
            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* →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

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((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)

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

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

Chart[0]

S0	$\gamma \rightarrow \bullet S$	[0,0]	[]	Dummy start state
S1	$S \rightarrow \bullet NP\ VP$	[0,0]	[]	Predictor
S2	$NP \rightarrow \bullet Det\ NOMINAL$	[0,0]	[]	Predictor
S3	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	[]	Predictor
S4	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	[]	Predictor
S5	$S \rightarrow \bullet VP$	[0,0]	[]	Predictor
S6	$VP \rightarrow \bullet Verb$	[0,0]	[]	Predictor
S7	$VP \rightarrow \bullet Verb\ NP$	[0,0]	[]	Predictor

Chart[1]

S8	$Verb \rightarrow book\bullet$	[0,1]	[]	Scanner
S9	$VP \rightarrow Verb\bullet$	[0,1]	[S8]	Completer
S10	$S \rightarrow VP\bullet$	[0,1]	[S9]	Completer
S11	$VP \rightarrow Verb \bullet NP$	[0,1]	[S8]	Completer
S12	$NP \rightarrow \bullet Det\ NOMINAL$	[1,1]	[]	Predictor
S13	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	[]	Predictor

Chart[2]

S14	$Det \rightarrow that\bullet$	[1,2]	[]	Scanner
S15	$NP \rightarrow Det\bullet NOMINAL$	[1,2]	[S14]	Completer
S16	$NOMINAL \rightarrow \bullet Noun$	[2,2]	[]	Predictor
S17	$NOMINAL \rightarrow \bullet Noun\ NOMINAL$	[2,2]	[]	Predictor

Chart[3]

S18	$Noun \rightarrow flight\bullet$	[2,3]	[]	Scanner
S19	$NOMINAL \rightarrow Noun\bullet$	[2,3]	[S18]	Completer
S20	$NOMINAL \rightarrow Noun\bullet NOMINAL$	[2,3]	[S18]	Completer
S21	$NP \rightarrow Det\ NOMINAL \bullet$	[1,3]	[S14,S19]	Completer
S22	$VP \rightarrow Verb\ NP \bullet$	[0,3]	[S8,S21]	Completer
S23	$S \rightarrow VP\bullet$	[0,3]	[S22]	Completer
S24	$NOMINAL \rightarrow \bullet Noun$	[3,3]	[]	Predictor
S25	$NOMINAL \rightarrow \bullet Noun\ NOMINAL$	[3,3]	[]	Predictor