Parsing Natural Language

Top Down Parsing: Earley Algorithm

Parsing Natural Language- Recap

- Parsing algorithms can be separated into two classes:

1. Algorithms that parse an input from tokens to the **S** symbol (Bottom-Up)
2. Algorithms that operate from Top-to-bottom (Top-Down)

Note: There are also algorithms which operate in both ways internally, the most well known is the „Top-Left-Corner-Parser“

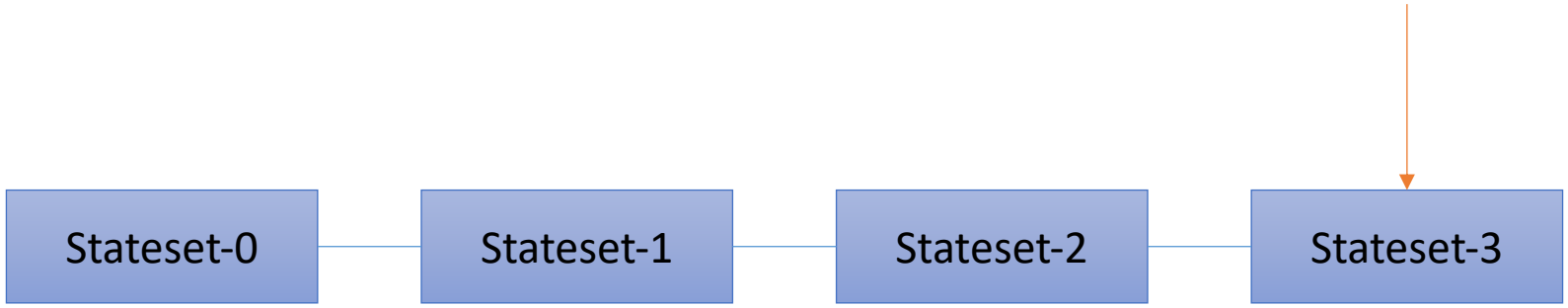
Earley Parser

- The Earley Parser is the most well known algorithm to parse input based on arbitrary! (no grammar conversion needed) Context free grammars
- It operates by repeatedly expanding partial parses into (hopefully) one parse which spans the entire input sequence
- The algorithm is yet again based on Dynamic Programming and therefore fills a clever data structure

Earley Parser – Data structure

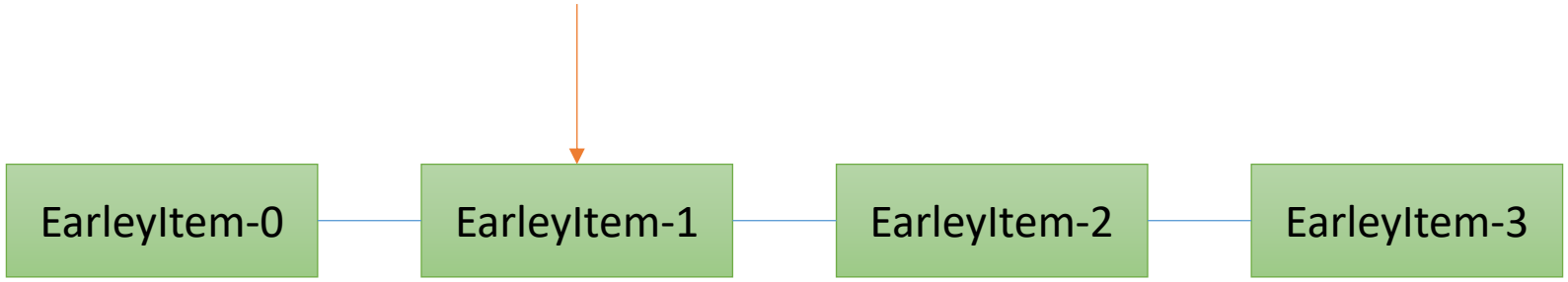
- Let us call this structure an „Earleychart“

- It is basically a container for a list of „state sets“, which additionally keeps track of the currently active set



Earley Parser – Data structure

- A state set in turn stores the progress of our parsing in the form of „EarleyItems“
  - And it also keeps track of the items that have already be considered



Earley Parser – Data structure

- An EarleyItem represents the progress of our parser
  - Progress is denoted in a „dot“ notation and an arbitrary CFG-rule
- Example Earley Item:

$NP \rightarrow$ WE ARE HERE● BUT HAVE NOT PARSED THIS [2]

This stands for: We found a potential! Noun phrase, starting in our input at position 2, however we can not be sure, since we only saw 3 items of the RHS, and there are still 5 left to produce

Earley Parser – Data structure

- The goal is to have an EarleyItem in our data structure, which
  1. Has a $S \rightarrow$
  2. Starts at the first input
  3. Is Completed
  4. Spans the entire input

S → WE ARE HERE AND HAVE PARSED THIS●[1,len(input)]

Earley Parser - Example

- Understanding the data structure means you already understood 90% of the algorithm

- Let us understand 100% of the algorithm by considering an example
  - And systematically fill the EarleyChart

Earley Parser - Example

- We are trying to parse the input:

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

- Using the CFG:

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- **Init**, start by looking at rules from $S \rightarrow$:

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0
$S \rightarrow \bullet NP\ OBJ$ [0]

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- Then loop through all items of the set

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0

$S \rightarrow \bullet NP\ OBJ$ [0]

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- Set the current item to active

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- Check the next symbol in this item

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- „Scan“ grammar, add everything starting with NP

Stateset-0

$$S \rightarrow \bullet NP\ OBJ\ [0]$$
$$NP \rightarrow \bullet V\ ADV\ NP\ [0]$$
$$NP \rightarrow \bullet DET\ NN\ [0]$$
$$NP \rightarrow \bullet NN\ [0]$$
$$NP \rightarrow \bullet DET\ ADJ\ NP\ [0]$$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- Continue with loop, first item is done!

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN\ [0]$
$NP \rightarrow \bullet NN\ [0]$
$NP \rightarrow \bullet DET\ ADJ\ NP\ [0]$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

- Scan again

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN\ [0]$
$NP \rightarrow \bullet NN\ [0]$
$NP \rightarrow \bullet DET\ ADJ\ NP\ [0]$
$V \rightarrow \bullet is\ [0]$
$V \rightarrow \bullet Climbing\ [0]$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- Continue loop

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN [0]$
$NP \rightarrow \bullet NN [0]$
$NP \rightarrow \bullet DET\ ADJ\ NP [0]$
$V \rightarrow \bullet is\ [0]$
$V \rightarrow \bullet Climbing\ [0]$
$DET \rightarrow \bullet a\ [0]$

$S \rightarrow NP\ OBJ$
$OBJ \rightarrow NP$
$NP \rightarrow V\ ADV\ NP$
$NP \rightarrow DET\ NN$
$NP \rightarrow NN$
$NP \rightarrow DET\ ADJ\ NP$

$V \rightarrow is$
$V \rightarrow Climbing$
$ADV \rightarrow down$
$DET \rightarrow a$
$ADJ \rightarrow superior$
$NN \rightarrow activity$
$NN \rightarrow tree$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- Continue loop

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN\ [0]$
$NP \rightarrow \bullet NN\ [0]$
$NP \rightarrow \bullet DET\ ADJ\ NP\ [0]$
$V \rightarrow \bullet is\ [0]$
$V \rightarrow \bullet Climbing\ [0]$
$DET \rightarrow \bullet a\ [0]$
$NN \rightarrow \bullet activity\ [0]$
$NN \rightarrow \bullet tree\ [0]$

$S \rightarrow NP\ OBJ$
$OBJ \rightarrow NP$
$NP \rightarrow V\ ADV\ NP$
$NP \rightarrow DET\ NN$
$NP \rightarrow NN$
$NP \rightarrow DET\ ADJ\ NP$

$V \rightarrow is$
$V \rightarrow Climbing$
$ADV \rightarrow down$
$DET \rightarrow a$
$ADJ \rightarrow superior$
$NN \rightarrow activity$
$NN \rightarrow tree$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- Continue loop

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN\ [0]$
$NP \rightarrow \bullet NN\ [0]$
$NP \rightarrow \bullet DET\ ADJ\ NP\ [0]$
$V \rightarrow \bullet is\ [0]$
$V \rightarrow \bullet Climbing\ [0]$
$DET \rightarrow \bullet a\ [0]$
$NN \rightarrow \bullet activity\ [0]$
$NN \rightarrow \bullet tree\ [0]$

$S \rightarrow NP\ OBJ$
$OBJ \rightarrow NP$
$NP \rightarrow V\ ADV\ NP$
$NP \rightarrow DET\ NN$
$NP \rightarrow NN$
$NP \rightarrow DET\ ADJ\ NP$

$V \rightarrow is$
$V \rightarrow Climbing$
$ADV \rightarrow down$
$DET \rightarrow a$
$ADJ \rightarrow superior$
$NN \rightarrow activity$
$NN \rightarrow tree$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

This would produce something we have in our set already (thats why it is a set – duh)
➔ We only need consider this if we have probabilities, since we want the max!

Earley Parser - Example

- Continue loop

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN\ [0]$
$NP \rightarrow \bullet NN\ [0]$
$NP \rightarrow \bullet DET\ ADJ\ NP\ [0]$
$V \rightarrow \bullet is\ [0]$
$V \rightarrow \bullet Climbing\ [0]$
$DET \rightarrow \bullet a\ [0]$
$NN \rightarrow \bullet activity\ [0]$
$NN \rightarrow \bullet tree\ [0]$

$S \rightarrow NP\ OBJ$
$OBJ \rightarrow NP$
$NP \rightarrow V\ ADV\ NP$
$NP \rightarrow DET\ NN$
$NP \rightarrow NN$
$NP \rightarrow DET\ ADJ\ NP$

$V \rightarrow is$
$V \rightarrow Climbing$
$ADV \rightarrow down$
$DET \rightarrow a$
$ADJ \rightarrow superior$
$NN \rightarrow activity$
$NN \rightarrow tree$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Finally something different! We found a terminal „is“, however our input pointer points to „Climbing“, so there is nothing we can do

Earley Parser - Example

- Continue loop

Stateset-0

$S \rightarrow \bullet NP\ OBJ$ [0]
$NP \rightarrow \bullet V\ ADV\ NP$ [0]
$NP \rightarrow \bullet DET\ NN$ [0]
$NP \rightarrow \bullet NN$ [0]
$NP \rightarrow \bullet DET\ ADJ\ NP$ [0]
$V \rightarrow \bullet is$ [0]
$V \rightarrow \bullet Climbing$ [0]
$DET \rightarrow \bullet a$ [0]
$NN \rightarrow \bullet activity$ [0]
$NN \rightarrow \bullet tree$ [0]

$S \rightarrow NP\ OBJ$
$OBJ \rightarrow NP$
$NP \rightarrow V\ ADV\ NP$
$NP \rightarrow DET\ NN$
$NP \rightarrow NN$
$NP \rightarrow DET\ ADJ\ NP$

$V \rightarrow is$
$V \rightarrow Climbing$
$ADV \rightarrow down$
$DET \rightarrow a$
$ADJ \rightarrow superior$
$NN \rightarrow activity$
$NN \rightarrow tree$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

This time it matches! This means this is a fruitful rule for further consideration

Stateset-1

$V \rightarrow Climbing \bullet$ [0]

Earley Parser - Example

- Continue loop

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-0

$S \rightarrow \bullet NP\ OBJ$ [0]
$NP \rightarrow \bullet V\ ADV\ NP$ [0]
$NP \rightarrow \bullet DET\ NN$ [0]
$NP \rightarrow \bullet NN$ [0]
$NP \rightarrow \bullet DET\ ADJ\ NP$ [0]
$V \rightarrow \bullet is$ [0]
$V \rightarrow \bullet Climbing$ [0]
$DET \rightarrow \bullet a$ [0]
$NN \rightarrow \bullet activity$ [0]
$NN \rightarrow \bullet tree$ [0]

No match for the rest ☹

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-1

$V \rightarrow Climbing \bullet$ [0]

Earley Parser - Example

- We completed state set 0!
- If there is an entry in the next state set, continue examining this state set
- Also move the pointer to the next input token

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0
$S \rightarrow \bullet NP\ OBJ$ [0]
$NP \rightarrow \bullet V\ ADV\ NP$ [0]
$NP \rightarrow \bullet DET\ NN$[0]
$NP \rightarrow \bullet NN$[0]
$NP \rightarrow \bullet DET\ ADJ\ NP$[0]
$V \rightarrow \bullet is$ [0]
$V \rightarrow \bullet climbing$ [0]
$DET \rightarrow \bullet a$ [0]
$NN \rightarrow \bullet activity$ [0]
$NN \rightarrow \bullet tree$ [0]

Stateset-1
$V \rightarrow Climbing \bullet$[0]

Earley Parser - Example

- Start loop again!

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-1

$V \rightarrow Climbing \bullet [0]$

$NP \rightarrow V \bullet ADV\ NP\ [0]$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

A completed item! This helps us to proceed some rules from previous state sets

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$

$NP \rightarrow \bullet V\ ADV\ NP\ [0]$

$NP \rightarrow \bullet DET\ NN [0]$

$NP \rightarrow \bullet NN [0]$

$NP \rightarrow \bullet DET\ ADJ\ NP [0]$

$V \rightarrow \bullet is\ [0]$

$V \rightarrow \bullet Climbing\ [0]$

$DET \rightarrow \bullet a\ [0]$

$NN \rightarrow \bullet activity\ [0]$

$NN \rightarrow \bullet tree\ [0]$

Earley Parser - Example

- Continue Loop

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-1

$V \rightarrow Climbing \bullet [0]$

$NP \rightarrow V \bullet ADV\ NP\ [0]$

$ADV \rightarrow \bullet down\ [1]$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0

$S \rightarrow \bullet NP\ OBJ\ [0]$

$NP \rightarrow \bullet V\ ADV\ NP\ [0]$

$NP \rightarrow \bullet DET\ NN [0]$

$NP \rightarrow \bullet NN [0]$

$NP \rightarrow \bullet DET\ ADJ\ NP [0]$

$V \rightarrow \bullet is\ [0]$

$V \rightarrow \bullet Climbing\ [0]$

$DET \rightarrow \bullet a\ [0]$

$NN \rightarrow \bullet activity\ [0]$

$NN \rightarrow \bullet tree\ [0]$

Earley Parser - Example

- Continue Loop

$S \rightarrow NP\ OBJ$
$OBJ \rightarrow NP$
$NP \rightarrow V\ ADV\ NP$
$NP \rightarrow DET\ NN$
$NP \rightarrow NN$
$NP \rightarrow DET\ ADJ\ NP$

$V \rightarrow is$
$V \rightarrow Climbing$
$ADV \rightarrow down$
$DET \rightarrow a$
$ADJ \rightarrow superior$
$NN \rightarrow activity$
$NN \rightarrow tree$

Stateset-1
$V \rightarrow Climbing\bullet[0]$
$NP \rightarrow V\bullet ADV\ NP\ [0]$
$ADV \rightarrow \bullet down\ [1]$

Stateset-2
$ADV \rightarrow down\bullet[1]$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-0
$S \rightarrow \bullet NP\ OBJ\ [0]$
$NP \rightarrow \bullet V\ ADV\ NP\ [0]$
$NP \rightarrow \bullet DET\ NN[0]$
$NP \rightarrow \bullet NN[0]$
$NP \rightarrow \bullet DET\ ADJ\ NP[0]$
$V \rightarrow \bullet is\ [0]$
$V \rightarrow \bullet Climbing\ [0]$
$DET \rightarrow \bullet a\ [0]$
$NN \rightarrow \bullet activity\ [0]$
$NN \rightarrow \bullet tree\ [0]$

Earley Parser - Example

- This goes on for a while…

- If there are no entries in next state sets, before we reached the end of our input, then we can never find a parse tree!

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-1
$V \rightarrow climbing\bullet$ [0]
$NP \rightarrow V\bullet ADV\ NP$ [0]
$ADV \rightarrow \bullet down$ [1]

Stateset-0
$S \rightarrow \bullet NP\ OBJ$ [0]
$NP \rightarrow \bullet V\ ADV\ NP$ [0]
$NP \rightarrow \bullet DET\ NN$ [0]
$NP \rightarrow \bullet NN$ [0]
$NP \rightarrow \bullet DET\ ADJ\ NP$ [0]
$V \rightarrow \bullet is$ [0]
$V \rightarrow \bullet Climbing$ [0]
$DET \rightarrow \bullet a$ [0]
$NN \rightarrow \bullet activity$ [0]
$NN \rightarrow \bullet tree$ [0]

Earley Parser - Example

- I'm just printing the finished state sets for you to check your understanding

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-2
ADV→down ● [1]
NP→V ADV ● NP [0]
NP→● V ADV NP [2]
NP→● DET NN [2]
NP→● NN [2]
NP→● DET ADJ NP [2]
V→● is [2]
V→● Climbing [2]
DET→● a [2]
NN→● activity [2]
NN→● tree [2]

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- I'm just printing the finished state sets for you to check your understanding

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Stateset-3
DET→a ● [2]
NP→DET ● NN [2]
NP→DET ● ADJ NP [2]
NN→● activity [3]
NN→● tree [3]
ADJ→● superior [3]

Earley Parser - Example

- I'm just printing the finished state sets for you to check your understanding

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-4
NN→tree ● [3]
NP→DET NN ● [2]
NP→V ADV NP ● [0]
S→NP ● V OBJ [0]
V→● is [4]
V→● Climbing [4]

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- I'm just printing the finished state sets for you to check your understanding

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-5
V→is ● [4]
S→NP V ● OBJ [0]
OBJ→● NP [5]
NP→● V ADV NP [5]
NP→● DET NN [5]
NP→● NN [5]
NP→● DET ADJ NP [5]
V→● is [5]
V→● Climbing [5]
DET→● a [5]
NN→● activity [5]
NN→● tree [5]

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- I'm just printing the finished state sets for you to check your understanding

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-6
DET→a ● [5]
NP→DET ● NN [5]
NP→DET ● ADJ NP [5]
NN→● activity [6]
NN→● tree [6]
ADJ→● superior [6]

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- I'm just printing the finished state sets for you to check your understanding

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

Stateset-7
ADJ→superior ● [6]
NP→DET ADJ ● NP [5]
NP→● V ADV NP [7]
NP→● DET NN [7]
NP→● NN [7]
NP→● DET ADJ NP [7]
V→● is [7]
V→● Climbing [7]
DET→● a [7]
NN→● activity [7]
NN→● tree [7]

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

Earley Parser - Example

- Once we have no more items to check, we can examine our result

$$S \rightarrow NP\ OBJ$$
$$OBJ \rightarrow NP$$
$$NP \rightarrow V\ ADV\ NP$$
$$NP \rightarrow DET\ NN$$
$$NP \rightarrow NN$$
$$NP \rightarrow DET\ ADJ\ NP$$

$$V \rightarrow is$$
$$V \rightarrow Climbing$$
$$ADV \rightarrow down$$
$$DET \rightarrow a$$
$$ADJ \rightarrow superior$$
$$NN \rightarrow activity$$
$$NN \rightarrow tree$$

[Climbing] [down] [a] [tree] [is] [a] [superior] [activity]

The goal is to have an EarleyItem in our data structure, which

1. Has a $S \rightarrow$
2. Starts at the first input
3. Is Completed
4. Spans the entire input

Stateset-8

$$S \rightarrow NP\ V\ OBJ \bullet\ [0]$$
$$OBJ \rightarrow NP \bullet\ [5]$$
$$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$$
$$NP \rightarrow NN \bullet\ [7]$$
$$NN \rightarrow activity \bullet\ [7]$$

Earley Parser - Example

- Okay so far, we could verify if the input is recognized using our grammar
- We have yet to build the tree from the data structure
- But let us state the algorithm first…

Earley Recognizer – Code stump

Algorithm: Earley Recognize

Input: G: Grammar, *terminals*: List<Terminal>

Output: Boolean, whether input was recognized with G

$chart = EarleyChart.empty()$

$chart[0].add(G.rulesWith(S \rightarrow))$ // init

Earley Recognizer – Code stump

Algorithm: Earley Recognize

Input: G: Grammar, *terminals*: List<Terminal>

Output: Boolean, whether input was recognized with G

$chart = EarleyChart.empty()$

$chart[0].add(G.rulesWith(S \rightarrow))$ // init

$while(chart.hasNext())$ // outerloop

$\quad currentState = chart.next()$

Earley Recognizer – Code stump

Algorithm: Earley Recognize

Input: G: Grammar, *terminals*: List<Terminal>

Output: Boolean, whether input was recognized with G

```
chart = EarleyChart.empty()
chart[0].add(G.rulesWith(S →))   // init
while(chart.hasNext())           // outerloop,
    currentState = chart.next()
    while(currentState.hasNext())    // innerloop
      currentItem = currentState.next()
      neededSymbol = currentItem.nextSymbol()
```

Earley Recognizer – Code stump

Algorithm: Earley Recognize

Input: G: Grammar, *terminals*: List<Terminal>

Output: Boolean, whether input was recognized with G

$chart = EarleyChart.empty()$

$chart[0].add(G.rulesWith(S \rightarrow))$ // init

$while(chart.hasNext())$ // outerloop,

$currentState = chart.next()$

$while(currentState.hasNext())$ // innerloop

$currentItem = currentState.next()$

$neededSymbol = currentItem.nextSymbol()$

updateChart(chart,currentItem,neededSymbol)

$return\ chart.hasAccepting()$

Earley Recognizer - Code stump

Procedure: updateChart

Input: chart: EarleyChart, item:EarleyItem, symbol:Terminal

```
match(symbol)
    Case NonTerminal: scan(chart,item)
    Case Terminal: predict(chart,item)
    Case None: complete(chart,item)
```

// Add new Earleyitems to the currently active chart, based on rules with symbol →…

// Compare current input symbol, and on match add an item to next state, create state if necessary

// Access all rules from state item.start that need symbol, and add them to the current state, with the dot shifted by 1 position

Getting the Parsetrees..

- Whenever we recognize an input sentence, we have yet to produce the parse trees that represent the parsing process
- Sadly this is by no means trivial…

The Parsetrees

- We start with our accepted rule

$$S \rightarrow NP\ V\ OBJ \bullet\ [0]$$

- Aside from when it started, we can also add where it ended

$$S \rightarrow NP\ V\ OBJ \bullet\ [0, 8]$$

The Parsetrees

$$S \rightarrow NP\ V\ OBJ \bullet\ [0, 8]$$

- In order to have the dot ● moved over OBJ, we have to have a rule which completed this

Stateset-8

$S \rightarrow NP\ V\ OBJ \bullet\ [0]$

$OBJ \rightarrow NP \bullet\ [5]$

$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$

$NP \rightarrow NN \bullet\ [7]$

$NN \rightarrow activity \bullet\ [7]$

- This tells us, that the OBJ started in state 5

The Parsetrees

$S \rightarrow NP\ V\ OBJ \bullet\ [0,8]$

$OBJ \rightarrow NP \bullet\ [5]$

- We can denote this…

$NP\ [5,8]$

- In order to have the dot moved over NP, we need to have one rule completed in 8 as well, and since it is a single symbol it need to start in 5 as well

Not you, you only span [7,8]

Stateset-8

$S \rightarrow NP\ V\ OBJ \bullet\ [0]$

$OBJ \rightarrow NP \bullet\ [5]$

$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$

$NP \rightarrow NN \bullet\ [7]$

$NN \rightarrow activity \bullet\ [7]$

The Parsetrees

- We can denote this…

$S \rightarrow NP\ V\ OBJ \bullet\ [0,8]$

NP [5,8]

DET [5, ?] ADJ [? , ?] NP [? , 8]

$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$

Stateset-8

$S \rightarrow NP\ V\ OBJ \bullet\ [0]$

$OBJ \rightarrow NP \bullet\ [5]$

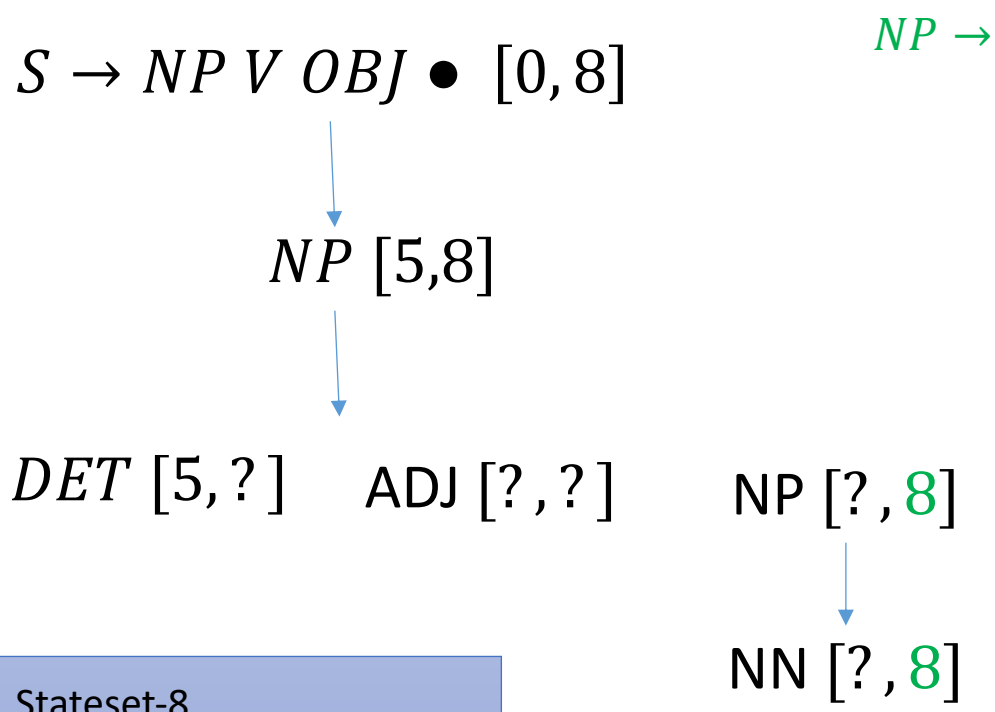
$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$

$NP \rightarrow NN \bullet\ [7]$

$NN \rightarrow activity \bullet\ [7]$

The Parsetrees

- We can denote this…



Stateset-8

$S \rightarrow NP\ V\ OBJ \bullet\ [0]$

$OBJ \rightarrow NP \bullet\ [5]$

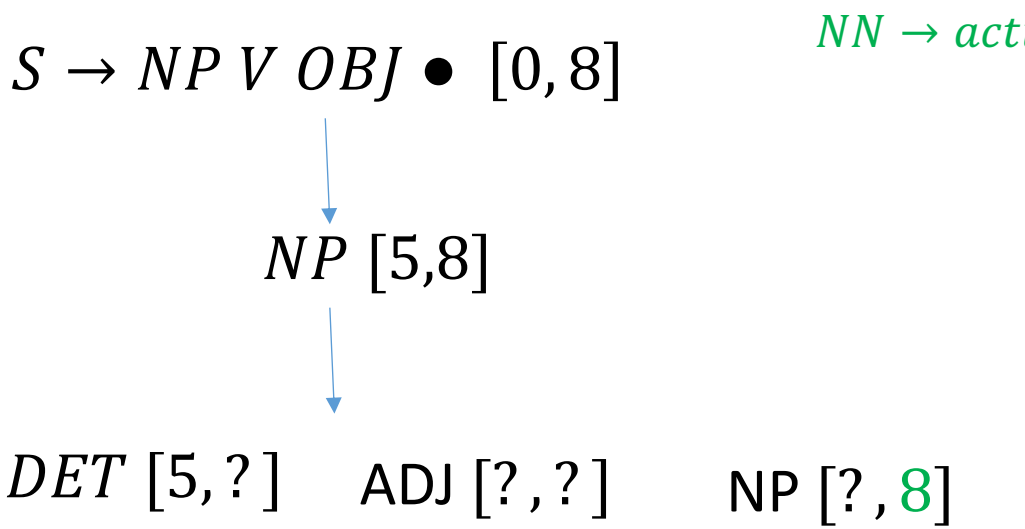
$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$

$NP \rightarrow NN \bullet\ [7]$

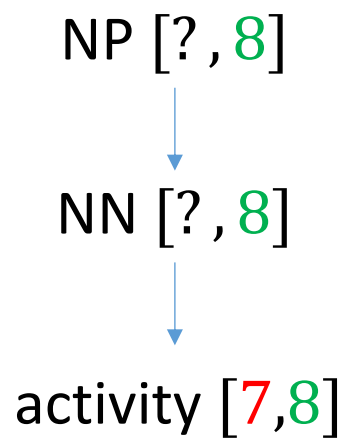
$NN \rightarrow activity \bullet\ [7]$

The Parsetrees

- We can denote this…



$NN \rightarrow activity \bullet\ [7]$

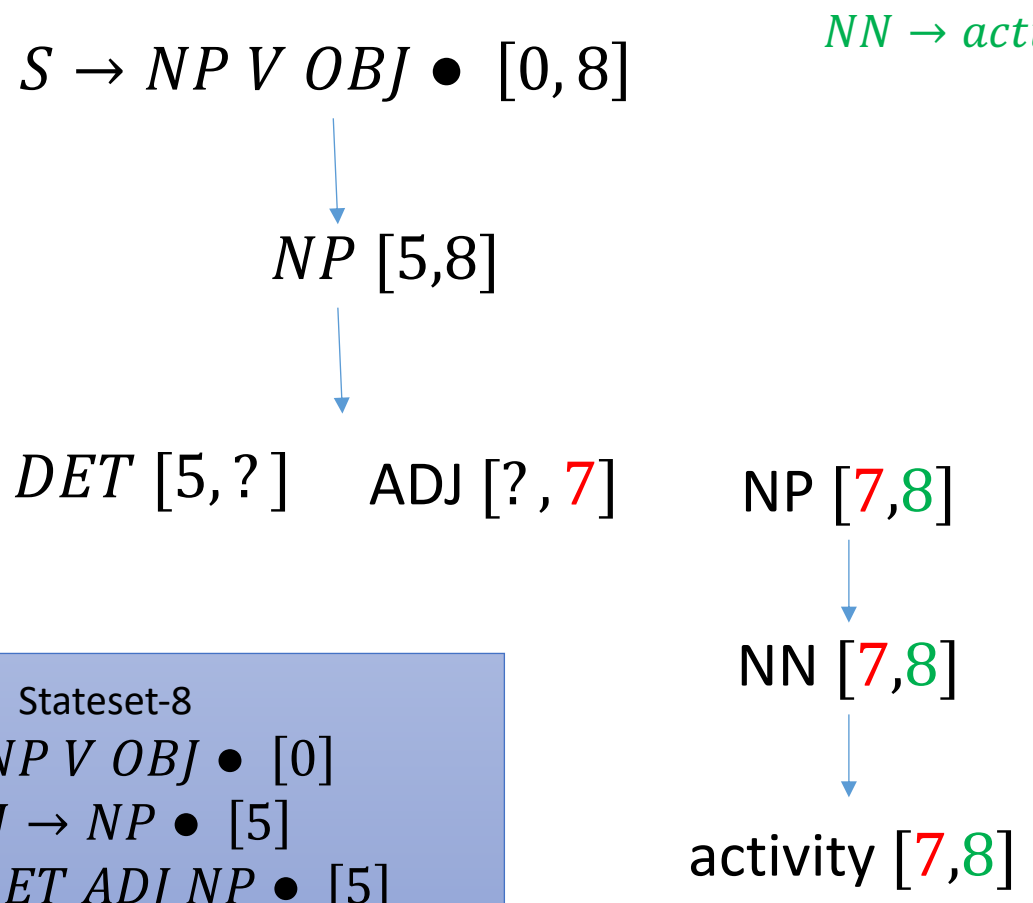


Stateset-8

$S \rightarrow NP\ V\ OBJ \bullet\ [0]$
$OBJ \rightarrow NP \bullet\ [5]$
$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$
$NP \rightarrow NN \bullet\ [7]$
$NN \rightarrow activity \bullet\ [7]$

The Parsetrees

- We can denote this…



Stateset-8

$S \rightarrow NP\ V\ OBJ \bullet\ [0]$

$OBJ \rightarrow NP \bullet\ [5]$

$NP \rightarrow DET\ ADJ\ NP \bullet\ [5]$

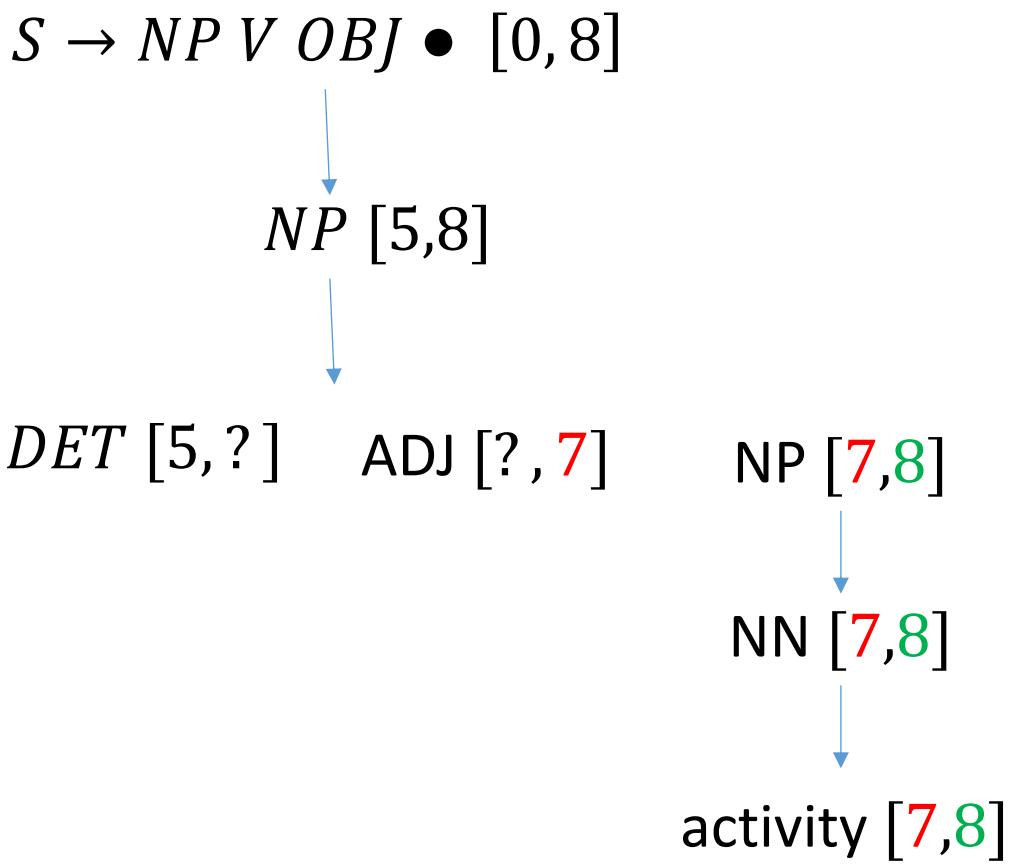
$NP \rightarrow NN \bullet\ [7]$

$NN \rightarrow activity \bullet\ [7]$

The Parsetrees

- We can denote this…

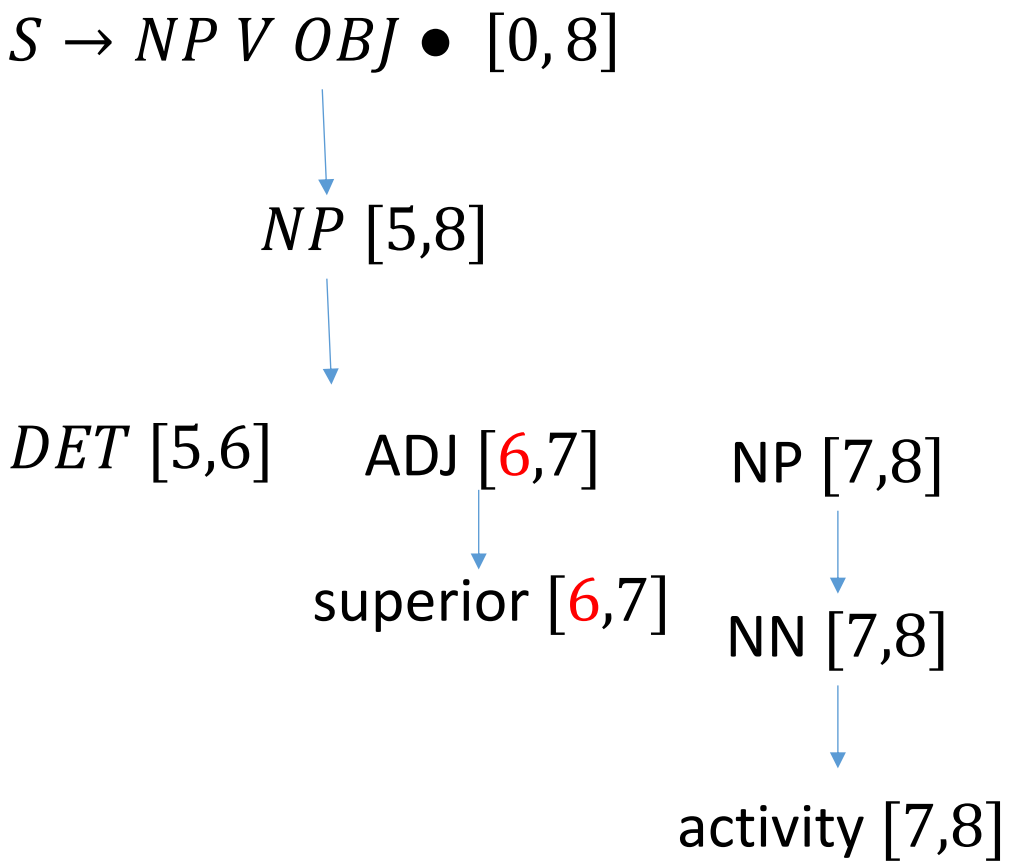
Stateset-7
ADJ→superior ● [6]
NP→DET ADJ ● NP [5]
NP→● V ADV NP [7]
NP→● DET NN [7]
NP→● NN [7]
NP→● DET ADJ NP [7]
V→● is [7]
V→● climbing [7]
DET→● a [7]
NN→● activity [7]
NN→● tree [7]



The Parsetrees

- We can denote this…

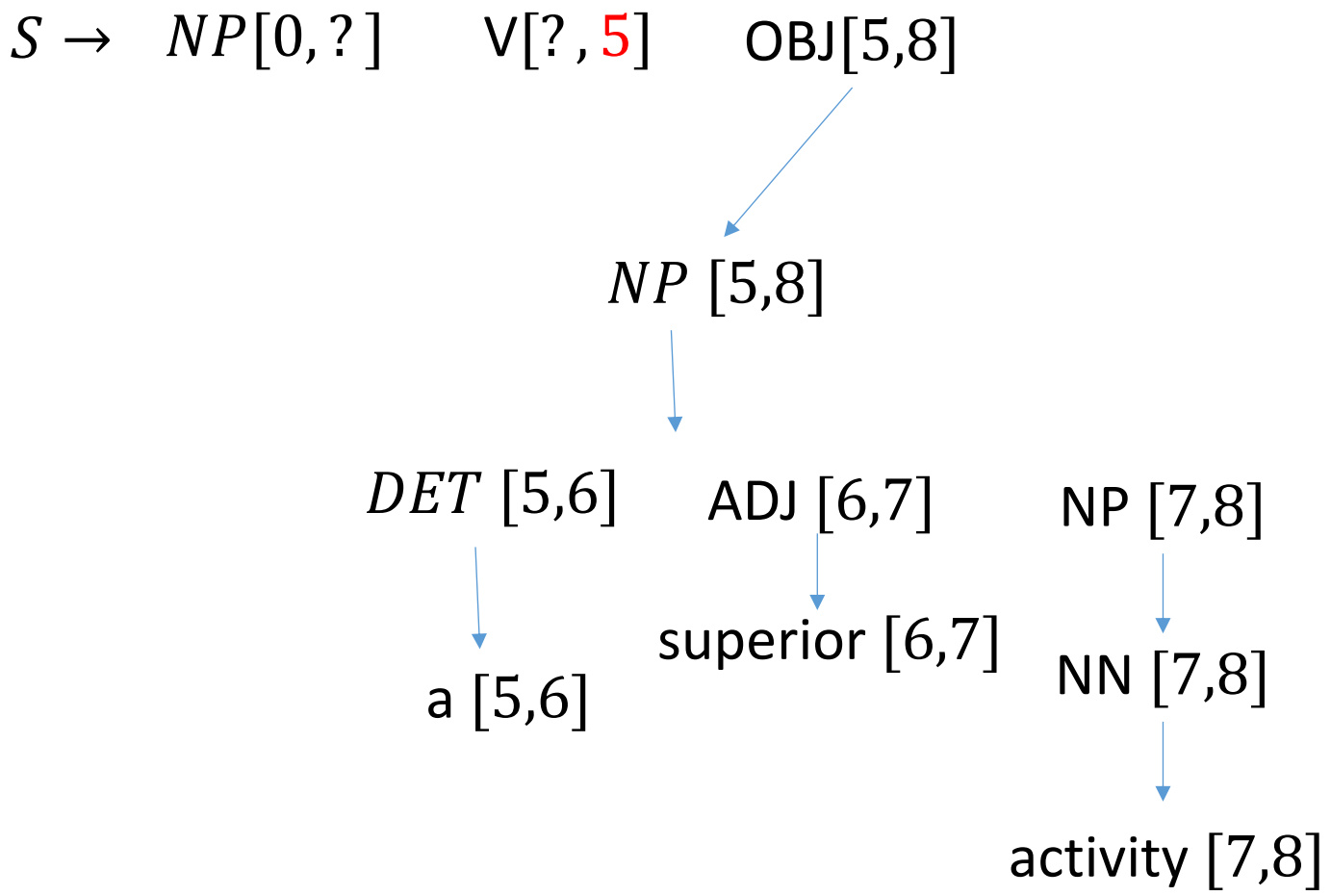
Stateset-7
ADJ→superior ● [6]
NP→DET ADJ ● NP [5]
NP→● V ADV NP [7]
NP→● DET NN [7]
NP→● NN [7]
NP→● DET ADJ NP [7]
V→● is [7]
V→● climbing [7]
DET→● a [7]
NN→● activity [7]
NN→● tree [7]



The Parsetrees

- We can denote this…

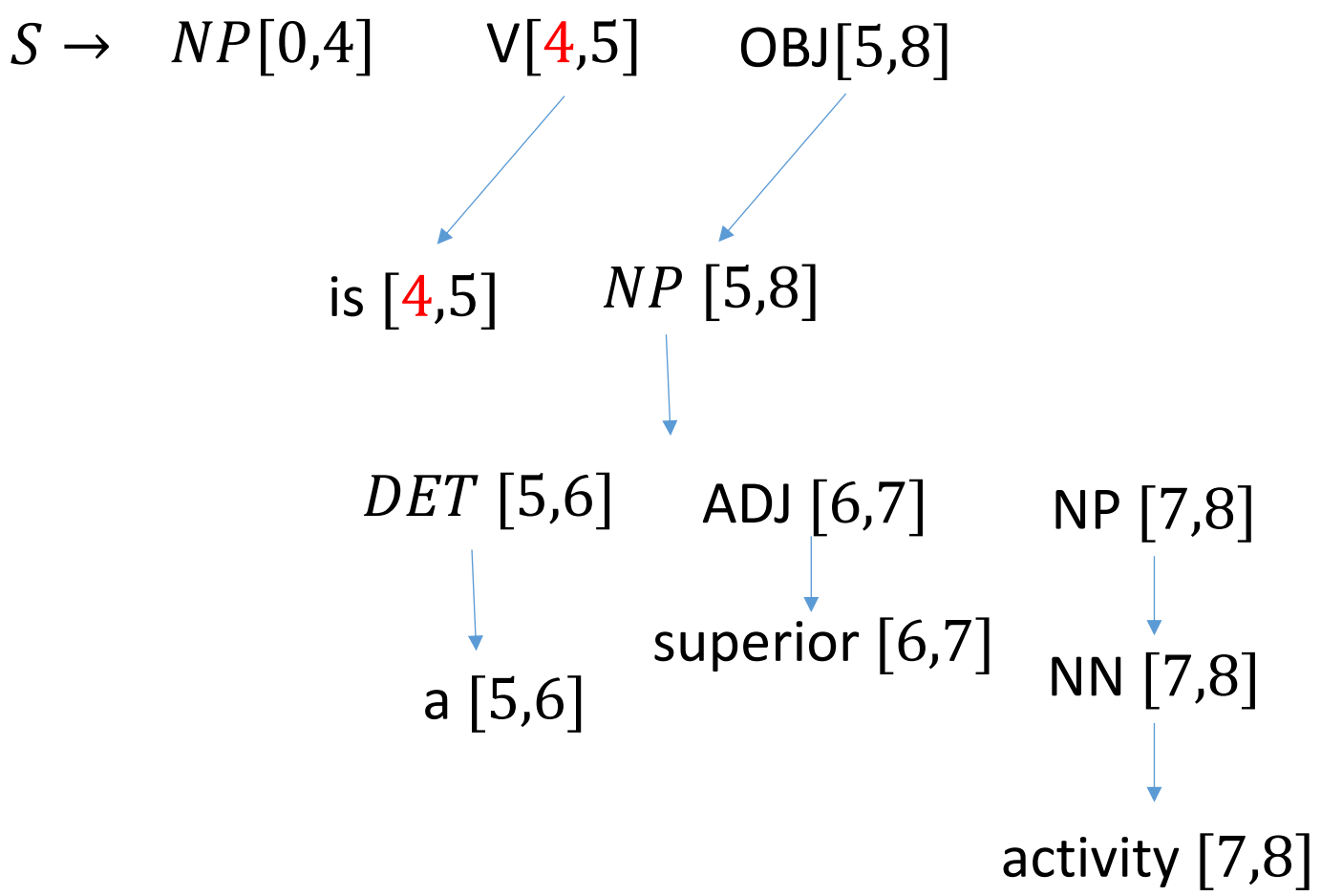
Stateset-6
DET→a ● [5]
NP→DET ● NN [5]
NP→DET ● ADJ NP [5]
NN→● activity [6]
NN→● tree [6]
ADJ→● superior [6]



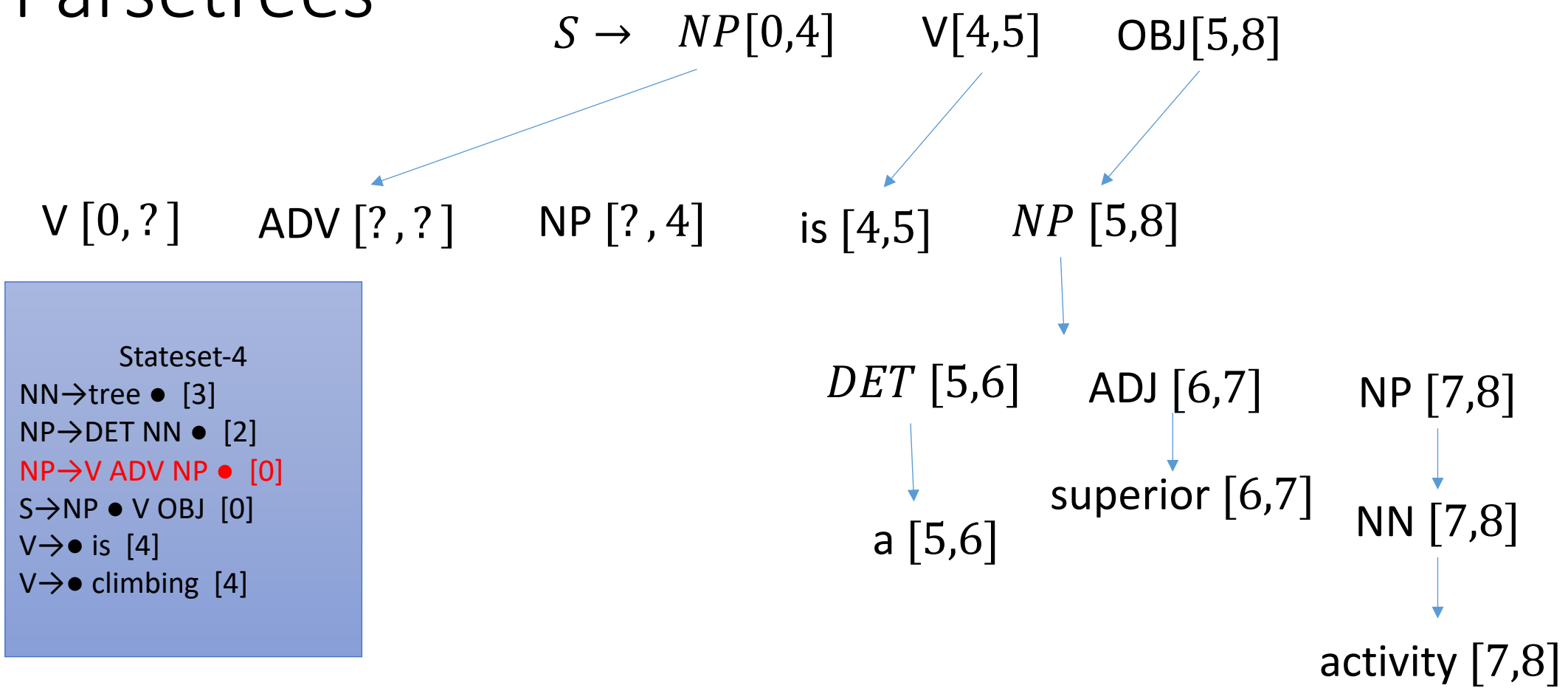
The Parsetrees

- We can denote this…

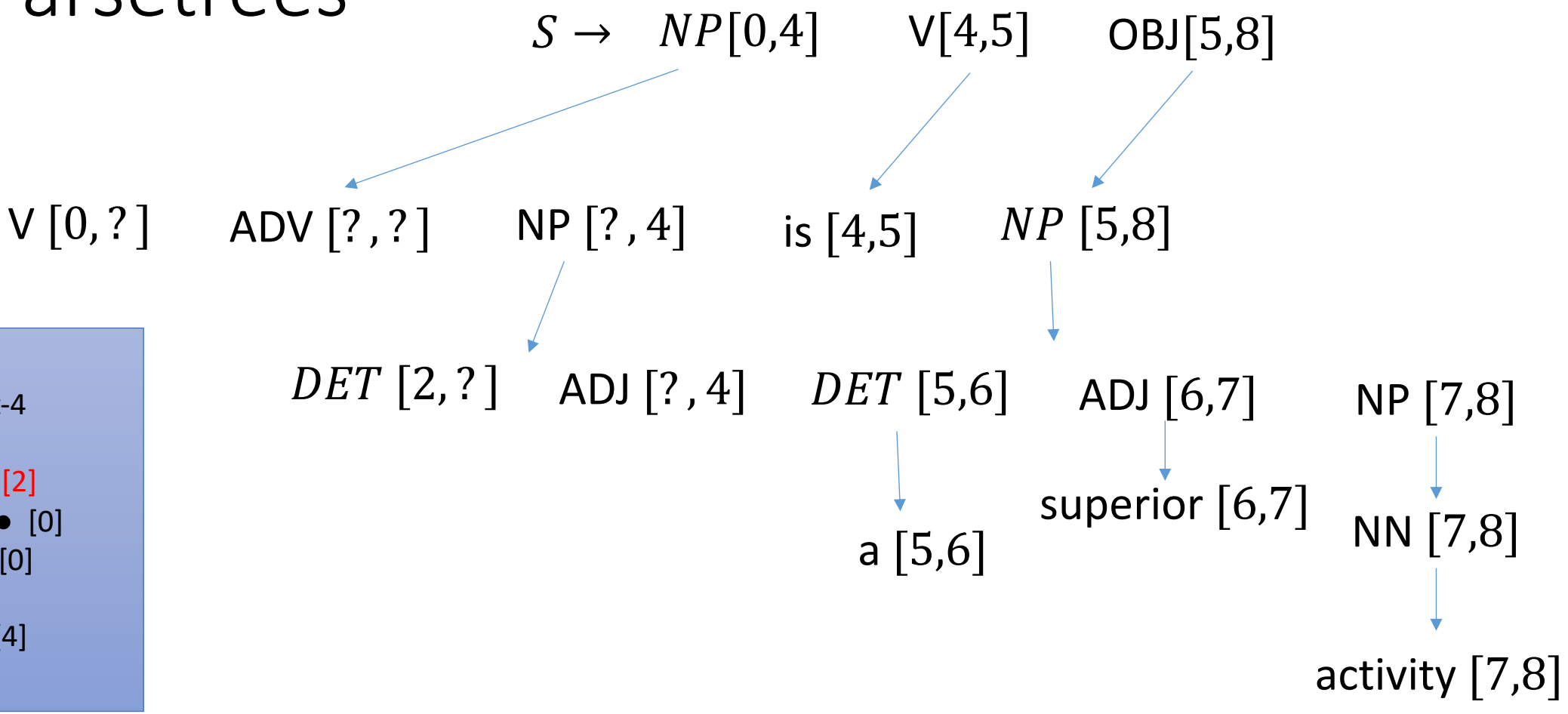
Stateset-5
V→is ● [4]
S→NP V ● OBJ [0]
OBJ→● NP [5]
NP→● V ADV NP [5]
NP→● DET NN [5]
NP→● NN [5]
NP→● DET ADJ NP [5]
V→● is [5]
V→● climbing [5]
DET→● a [5]
NN→● activity [5]
NN→● tree [5]



The Parsetrees

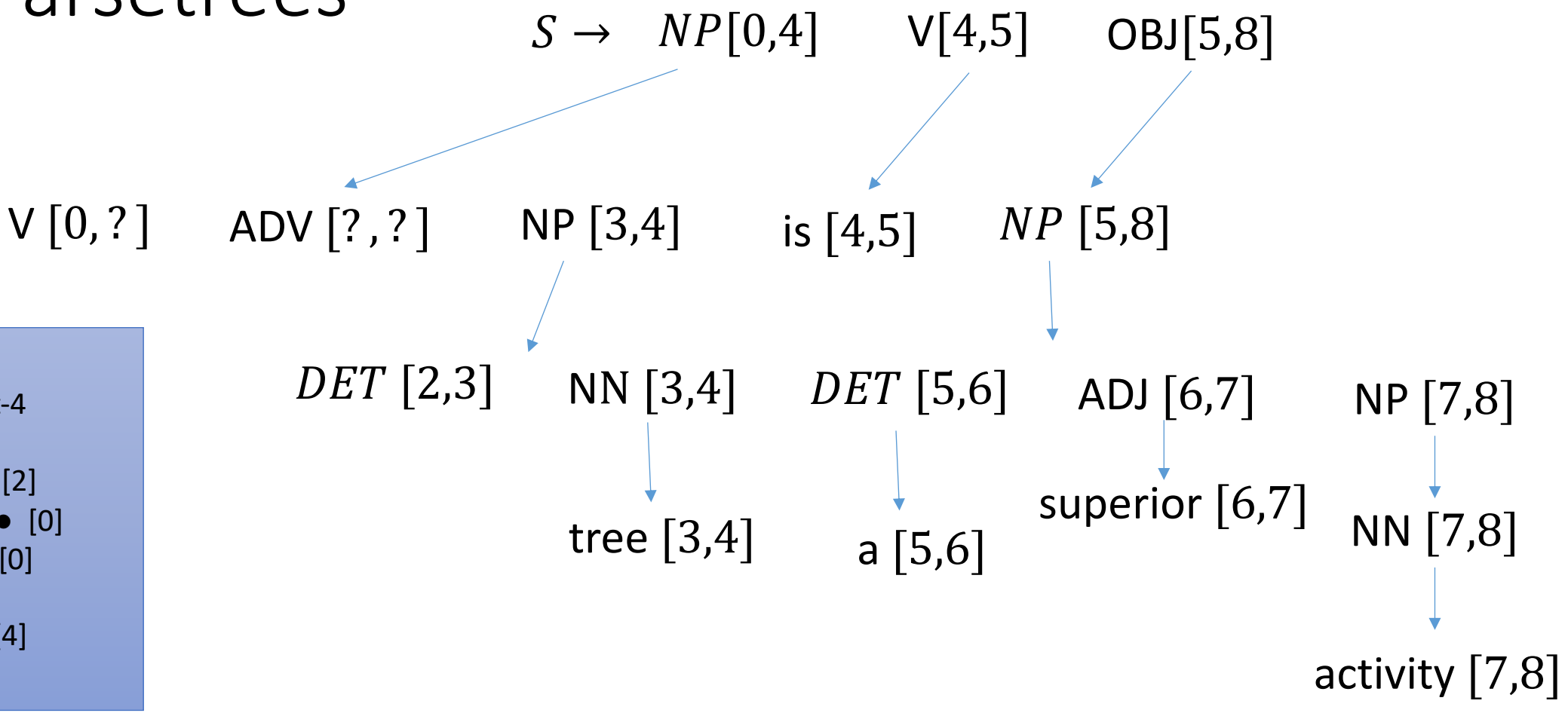


The Parsetrees



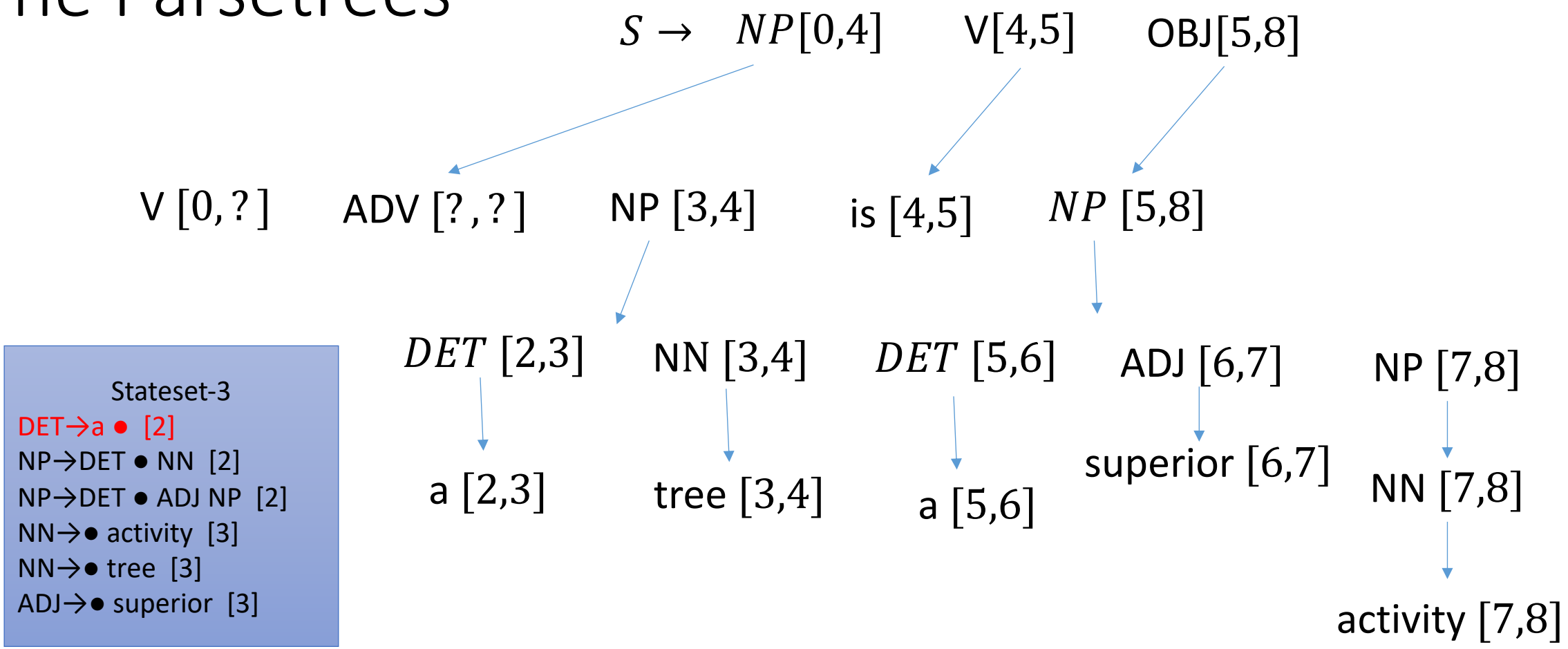
Stateset-4
NN→tree ● [3]
NP→DET NN ● [2]
NP→V ADV NP ● [0]
S→NP ● V OBJ [0]
V→● is [4]
V→● climbing [4]

The Parsetrees

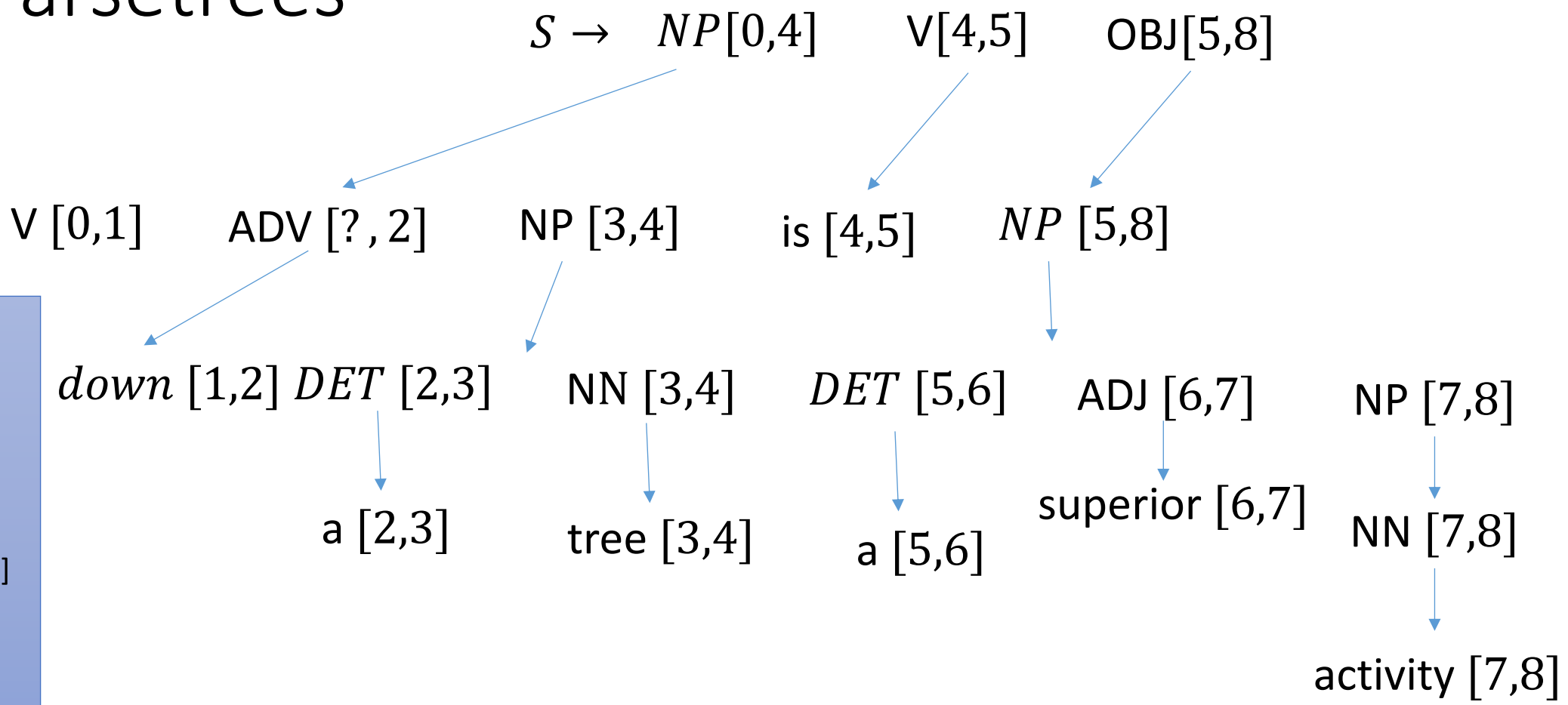


Stateset-4
NN→tree ● [3]
NP→DET NN ● [2]
NP→V ADV NP ● [0]
S→NP ● V OBJ [0]
V→● is [4]
V→● climbing [4]

The Parsetrees

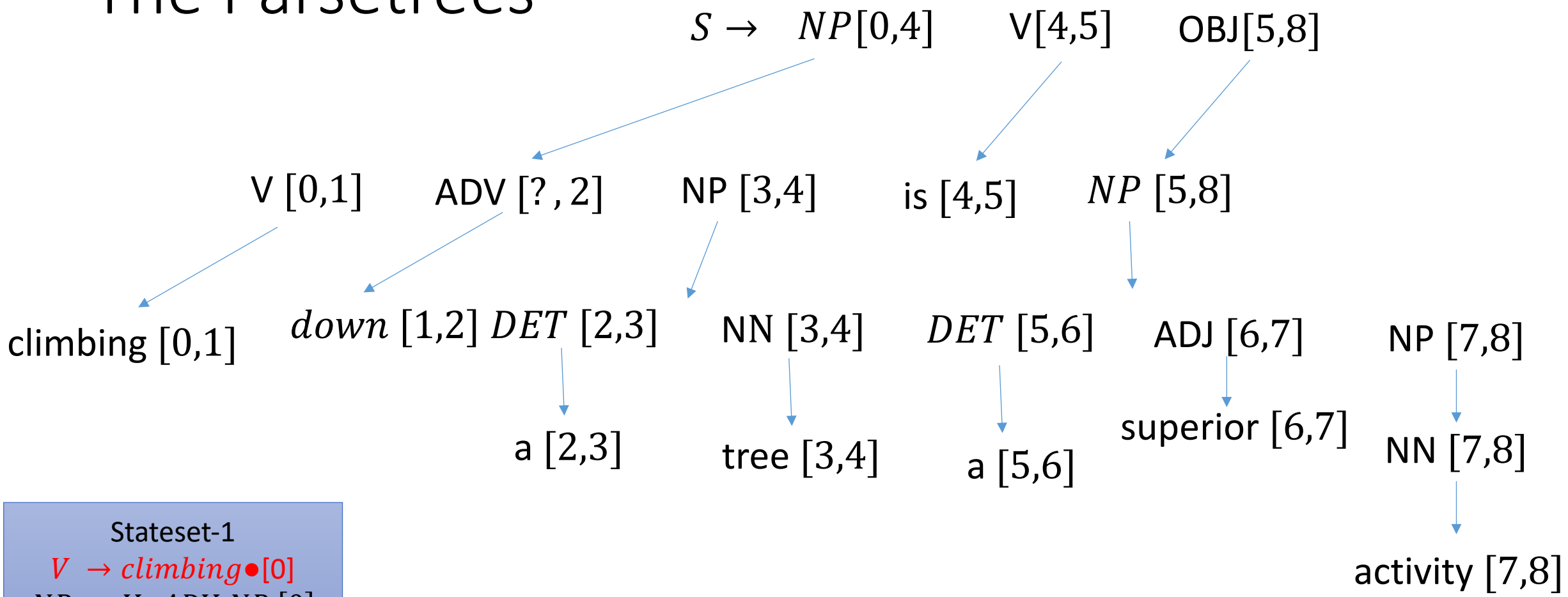


The Parsetrees



Stateset-2
ADV→down ● [1]
NP→V ADV ● NP [0]
NP→● V ADV NP [2]
NP→● DET NN [2]
NP→● NN [2]
NP→● DET ADJ NP [2]
V→● is [2]
V→● climbing [2]
DET→● a [2]
NN→● activity [2]
NN→● tree [2]

The Parsetrees



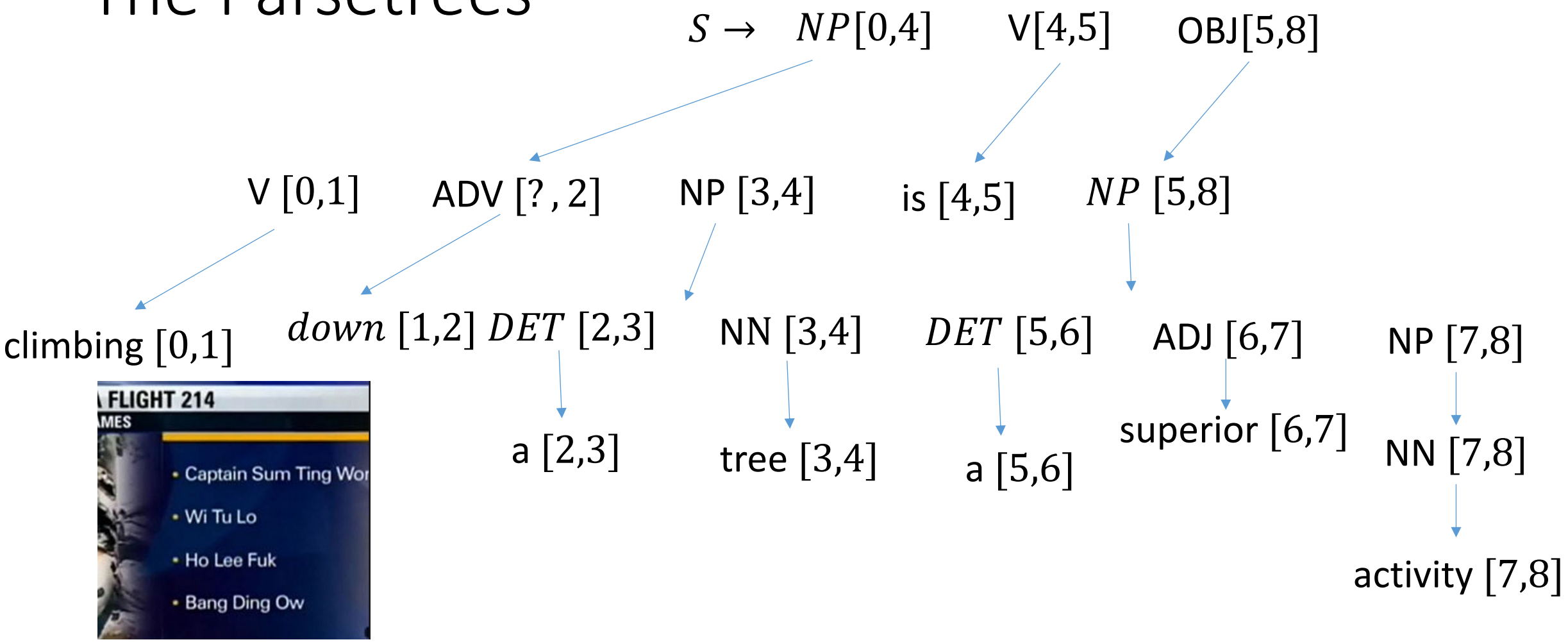
Stateset-1

$V \rightarrow climbing \bullet$[0]

$NP \rightarrow V \bullet ADV\ NP$ [0]

$ADV \rightarrow \bullet down$ [1]

The Parsetrees



The Parsetrees

- In fact building the parse trees is yet again a similar to parsing, which resembles a backtracking search over some partitions built in the „forward parse“ (the recognizer)
  - Called an „Unger-Parser“

- Also you usually do not build Parsetrees, but a „Parseforest“, which is an efficient data structure for dealing with „Parsetrees“

Recap Earley

- Earley is very generic algorithm for parsing sentences with a CFG
- Works from left-to-right, top-to-bottom
- No need for grammar conversion
- Runtime O(n^3)
  - Current knowledge limits this to the runtime of matrix multiplication, so the current fastest generic parsing algorithm is somewhere around $O(n^{2.3})$