Parsing

Parsing

- Parsing with CFGs refers to the task of assigning correct trees to input strings
- Correct here means a tree that covers all and only the elements of the input and has an S at the top
- It doesn't actually mean that the system can select the correct tree from among all the possible trees

Parsing

As with everything of interest, parsing involves a search which involves the making of choices

We’ll start with some basic (meaning bad) methods before moving on to the one or two that you need to know

Programming languages

```
printf ("/charset [%s",
        (re_opcode_t) *(p - 1) == charset_not ? "^" : "");

assert (p + *p < pend);

for (c = 0; c < 256; c++)
  if (c / 8 < *p && (p[1 + (c/8)] & (1 << (c % 8)))) {
     /* Are we starting a range?  */
     if (last + 1 == c && ! inrange) {
        putchar ('-');
        inrange = 1;
     }
     /* Have we broken a range?  */
     else if (last + 1 != c && inrange) {
        putchar (last);
        inrange = 0;
     }

     if (! inrange)
        putchar (c);

     last = c;
  }
```

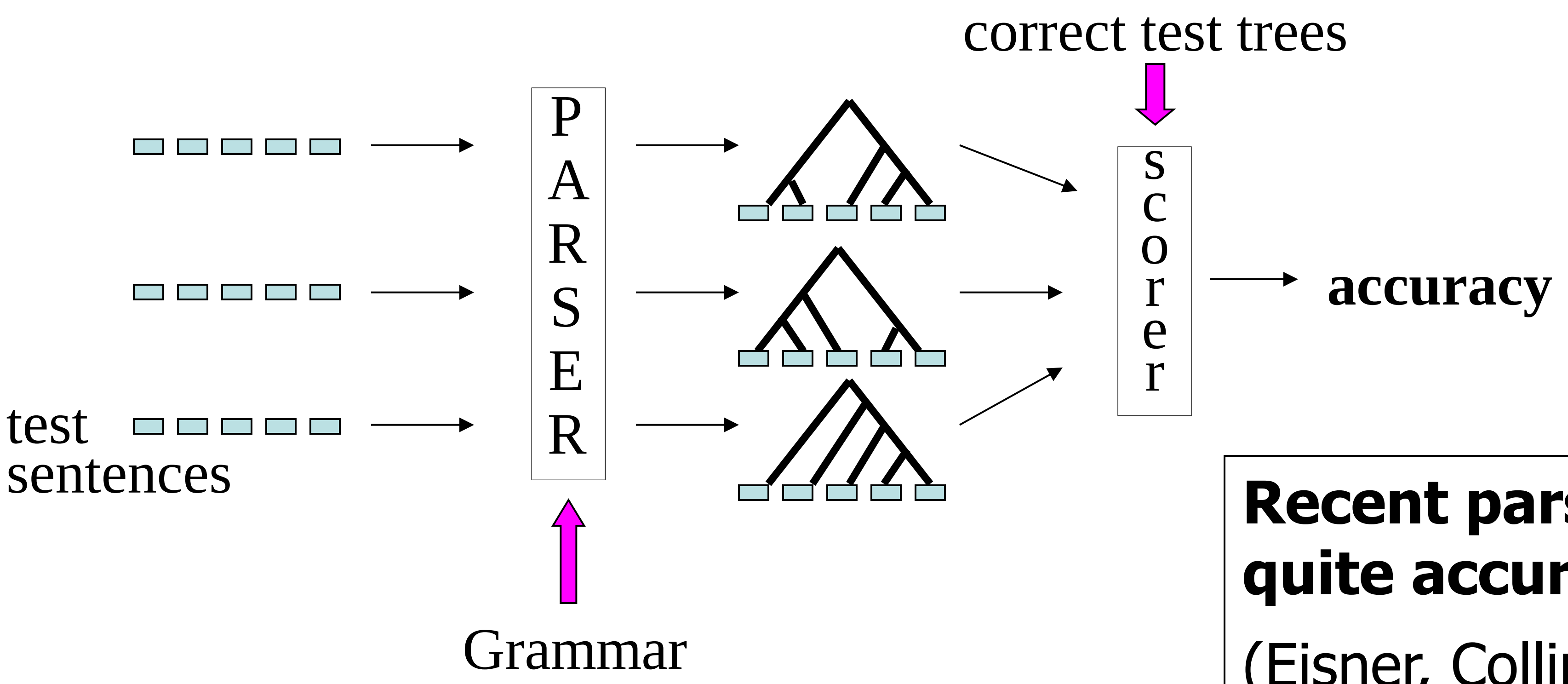
- Easy to parse.
- Designed that way!

Natural languages

```
printf "/charset %s", re_opcode_t *p - 1 == charset_not ? "^"
: ""; assert p + *p < pend; for c = 0; c < 256; c++ if c / 8 <
*p && p1 + c/8 & 1 << c % 8 Are we starting a range? if last +
1 == c && !  inrange putchar '-'; inrange = 1; Have we broken a
range? else if last + 1 != c && inrange putchar last; inrange =
0; if !  inrange putchar c; last = c;
```

- No {} () [] to indicate scope & precedence
- Lots of overloading (arity varies)
- Grammar isn’t known in advance!
- Context-free grammar not best formalism

The parsing problem

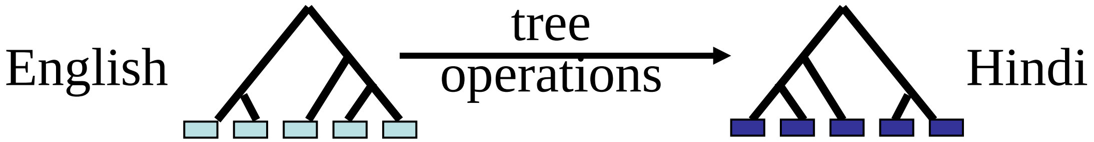


Recent parsers quite accurate

(Eisner, Collins, Charniak, etc.)

Applications of parsing (1/2)

- Machine translation (Alshawi 1996, Wu 1997, ...)



- Speech synthesis from parses (Prevost 1996)

  The government plans to raise income tax.
  The government plans to raise income tax the imagination.

- Speech recognition using parsing (Chelba et al 1998)

  Put the file in the folder.

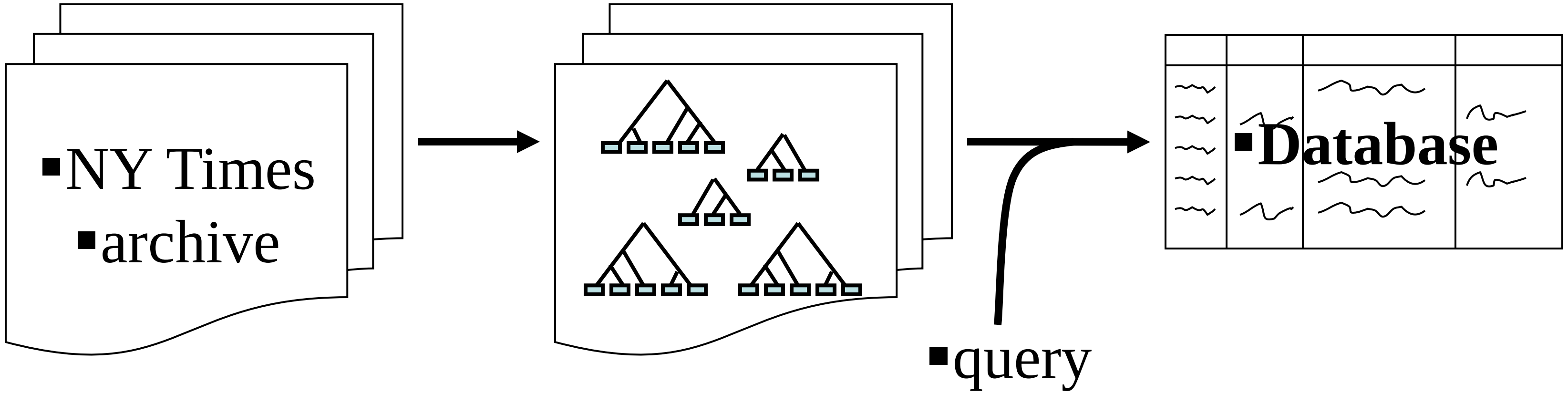
  Put the file and the folder.

Applications of parsing (2/2)

- Grammar checking (Microsoft)
- Indexing for information retrieval (Woods 1997)

  ... washing a car with a hose ... → vehicle maintenance

- Information extraction (Hobbs 1996)

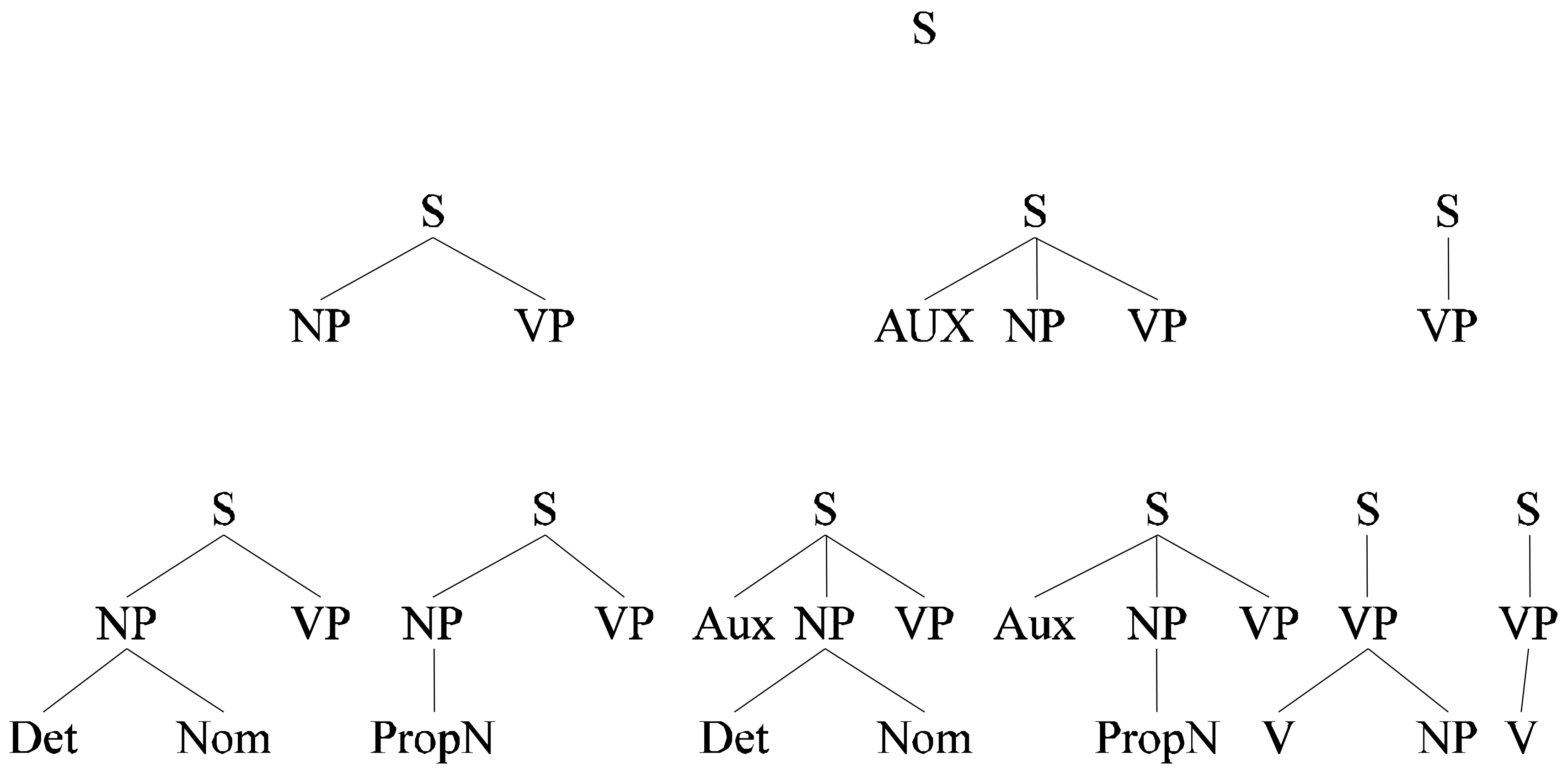


Top-Down Parsing

Since we're trying to find trees rooted with an S (Sentences) start with the rules that give us an S.

Then work your way down from there to the words.

Top Down Space

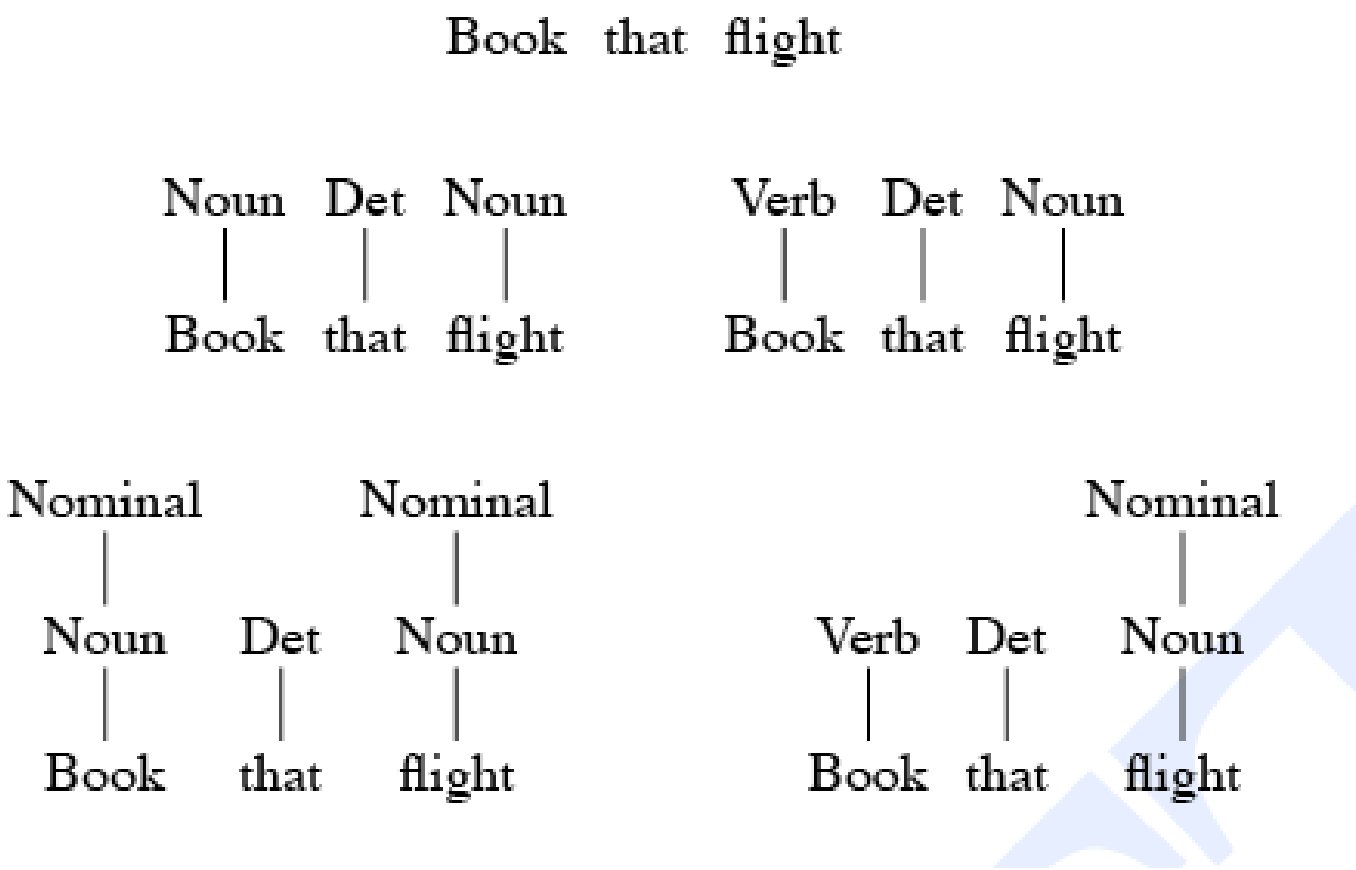


Bottom-Up Parsing

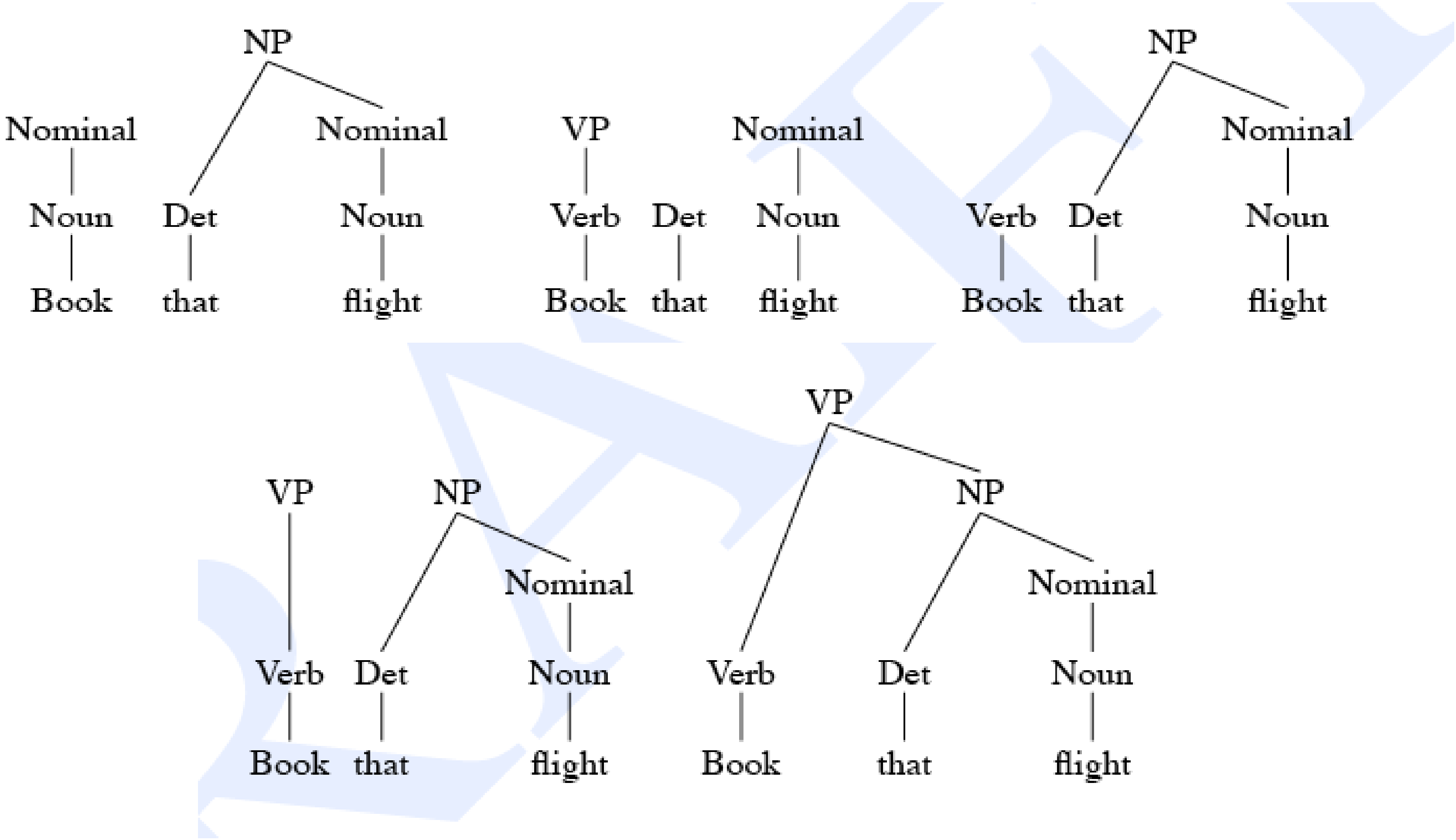
Of course, we also want trees that cover the input words. So start with trees that link up with the words in the right way.

Then work your way up from there.

Bottom-Up Space



Bottom Up Space



Control

- Of course, in both cases we left out how to keep track of the search space and how to make choices
  - Which node to try to expand next
  - Which grammar rule to use to expand a node

Top-Down and Bottom-Up

Top-down

Only searches for trees that can be answers (i.e. S’s)

But also suggests trees that are not consistent with any of the words

Bottom-up

Only forms trees consistent with the words

But suggest trees that make no sense globally

Problems

- Even with the best filtering, backtracking methods are doomed if they don’t address certain problems
  - Ambiguity

Ambiguity

I saw a man with the telescope.

Stolen painting found by the tree

Parsing

- Chart Parsing
  - **CKY**
  - **Earley**
- Both are dynamic programming solutions that run in O(n**3) time.
  - CKY is bottom-up
  - Earley is top-down

Top-Down Parsing

- DFS on the AND-OR graph

- Data structures:
  - *Open List (OL)*: Nodes to be expanded
  - *Closed List (CL)*: Expanded Nodes
  - *Input List (IL)*: Words of sentence to be parsed
  - *Moving Head (MH)*: Walks over the IL

Toy Grammar/ Fragment of Grammar

S → NP VP

VP → V NP

NP → N | ART N

N → Ram

V → ate | saw

Det → a | an | the

N → rice | apple | movie

Trace of Top-Down Parsing

Initial Condition (T_0)

S

OL

CL (empty)

Ram ate the rice

IL

MH

Trace of Top-Down Parsing

T_1:

NP VP		OL
S	CL	
Ram ate the rice	IL	

MH

Trace of Top-Down Parsing

T_2:

N Det N VP

OL

S NP

CL

Ram ate the rice

IL

MH

Trace of Top-Down Parsing

T_3:

Det N VP

OL

S NP N

CL

Ram ate the rice

IL

MH (portion of Input consumed)

Trace of Top-Down Parsing

T_4:

N VP	OL
S NP N Det*	CL
Ram ↑ ate the rice	IL

MH

(* indicates 'useless' expansion)

Trace of Top-Down Parsing

T_5:

VP	OL
S NP N Det* N*	CL
Ram ↑ ate the rice	IL

MH

Trace of Top-Down Parsing

T_6:

V NP	OL
S NP N Det* N*	CL
Ram ↑ ate the rice	IL

MH

Trace of Top-Down Parsing

T_7:

NP	OL
S NP N Det* N* V	CL
Ram ate the rice	IL

MH

Trace of Top-Down Parsing

T_8:

N Det N

OL

S NP N Det* N* V NP

CL

Ram ate the rice

IL

MH

Trace of Top-Down Parsing

T_9:

Det N

OL

S NP N Det* N* V N*

CL

Ram ate the rice

IL

MH

Trace of Top-Down Parsing

T_{10}:

N

OL

S NP N Det* N* V N* Det

CL

Ram ate the rice

IL

MH

Trace of Top-Down Parsing

T_{11}:

OL

S NP N Det* N* V N* Det N

CL

Ram ate the rice

IL

MH

Successful Termination: OL empty AND MH at the end of IL.

Bottom-Up Parsing

Basic idea:

- Refer to words from the lexicon.
- Obtain all POSs for each word.
- Keep combining until *S* is obtained.

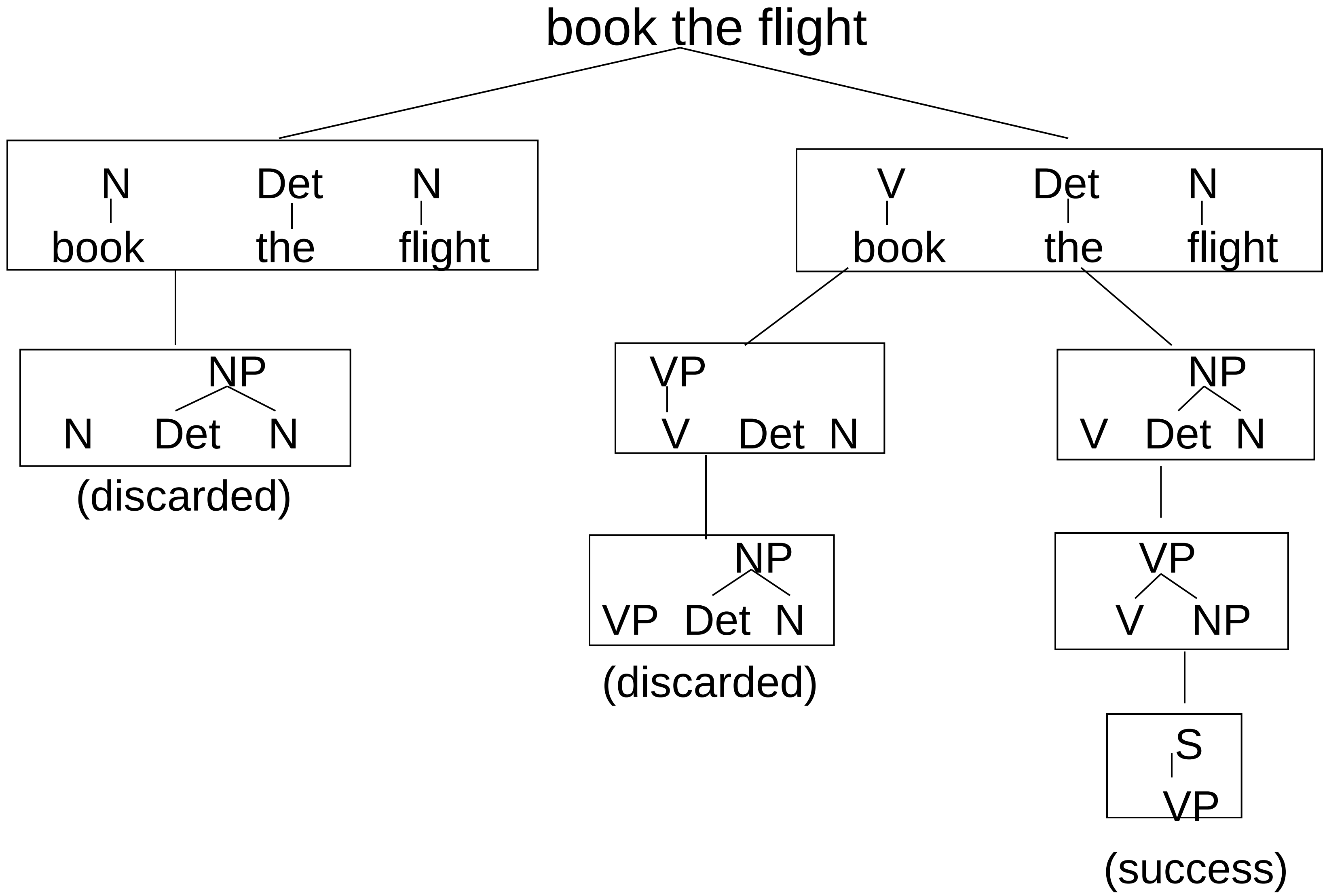
Example of Bottom-Up Parsing

Example from *Jurafsky & Martin, 2000 :*

- Sentence: Book the flight
- Grammar:

S	→	NP VP | VP
NP	→	Det N | N
VP	→	V | V NP
Det	→	a | an | the
N	→	book | flight | meal
V	→	book | include

Trace of Bottom-Up Parsing



Implementation of Bottom-up Parsing

Through a stack

Push words into the stack

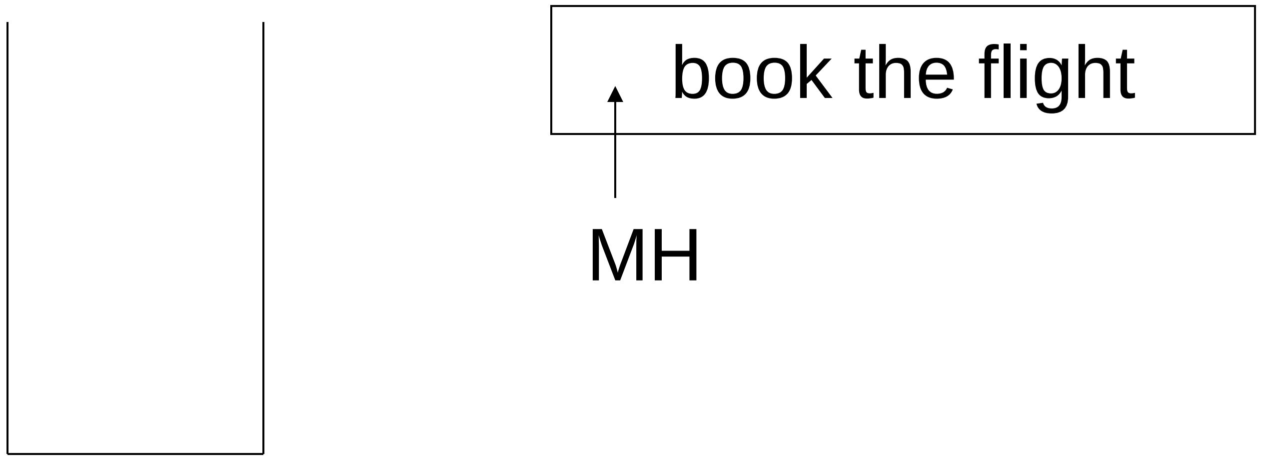
Look for a “handle” to reduce to a non-terminal

(Aho, Sethi, Ullman 1986)

Termination by “start symbol on stack” and “end of sentence”.

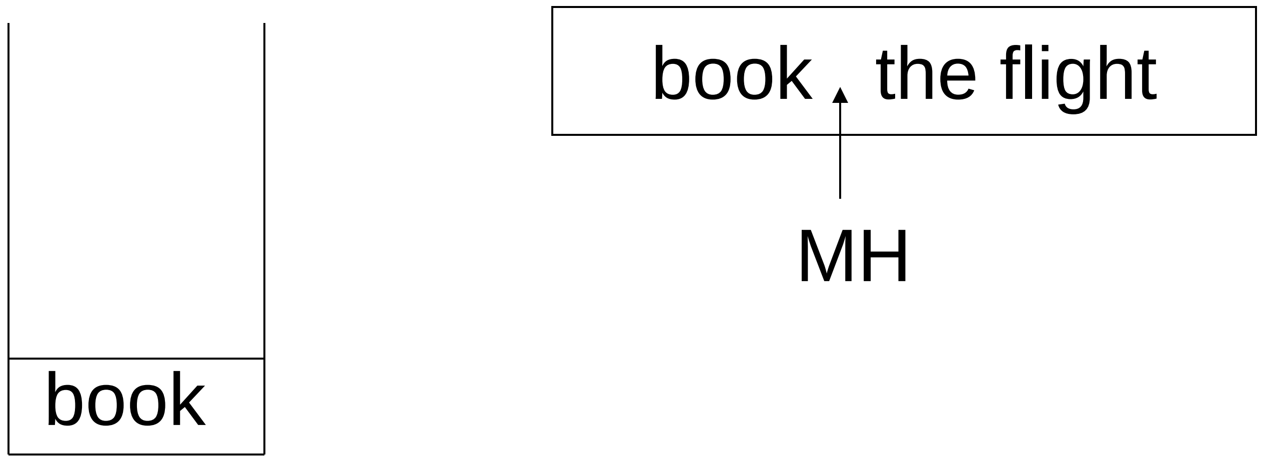
Trace of Bottom-Up Parsing

T_0



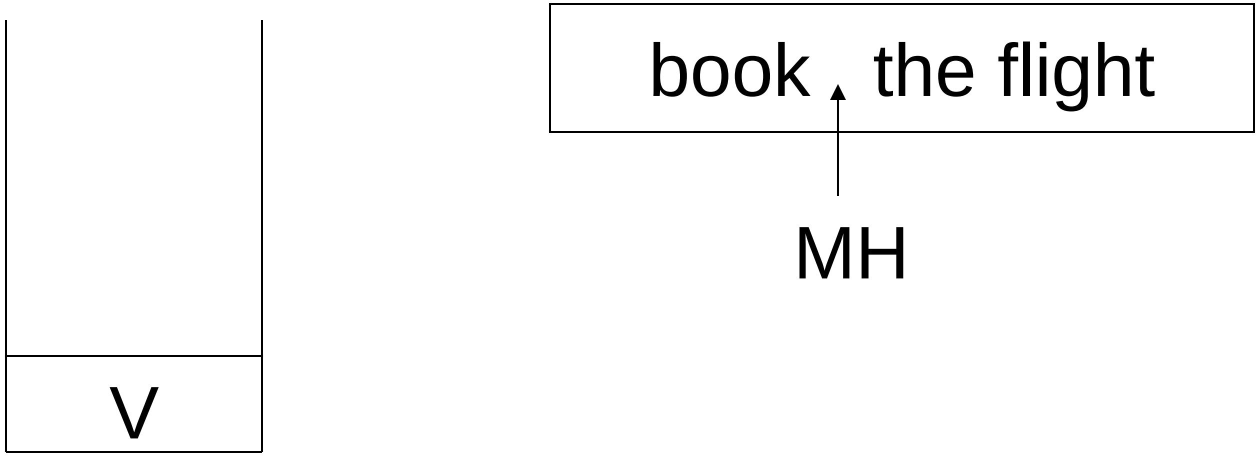
Trace of Bottom-Up Parsing

- Push ‘book’; advance input pointer



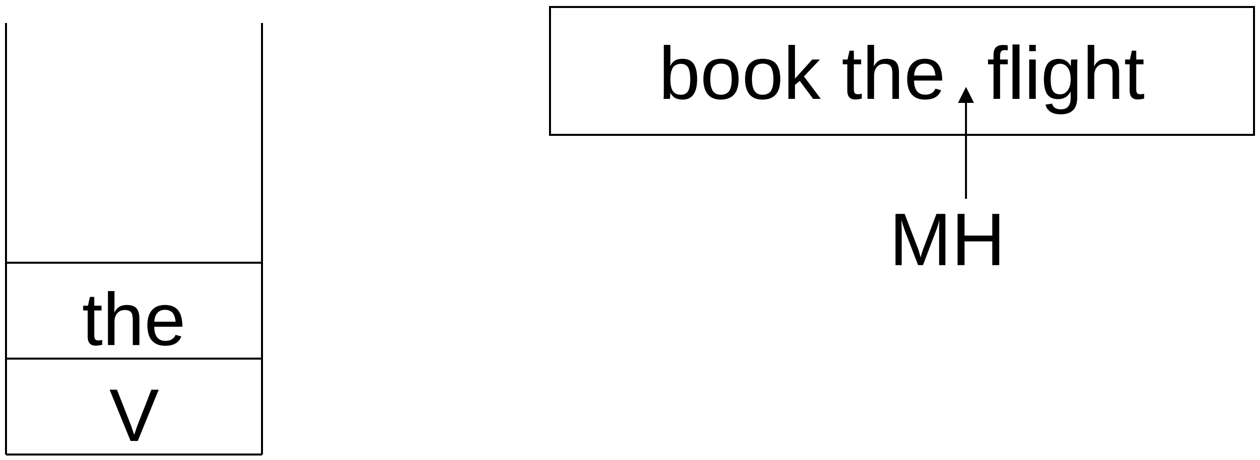
Trace of Bottom-Up Parsing

- Reduce ‘book’



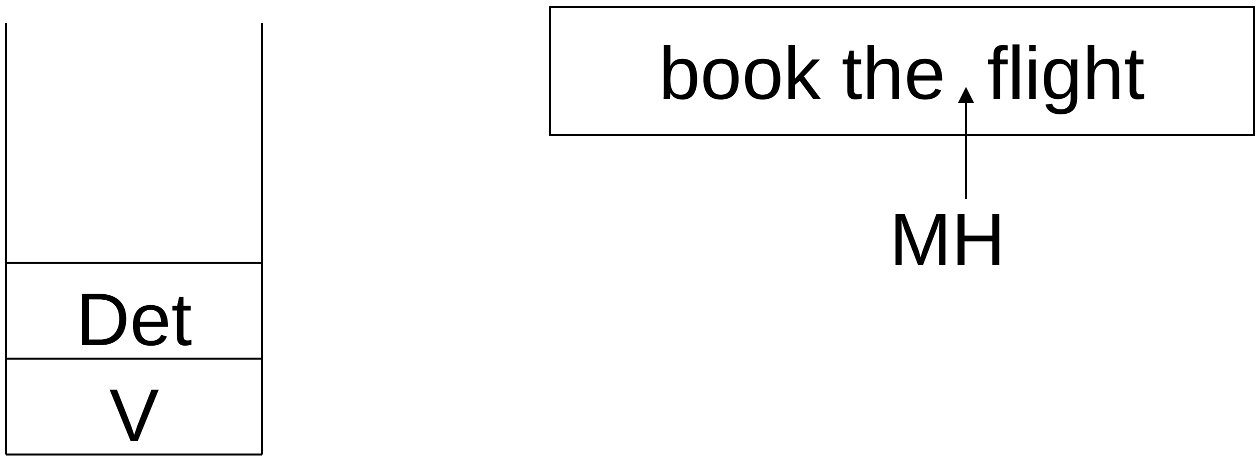
Trace of Bottom-Up Parsing

- Push ‘the’; advance input pointer



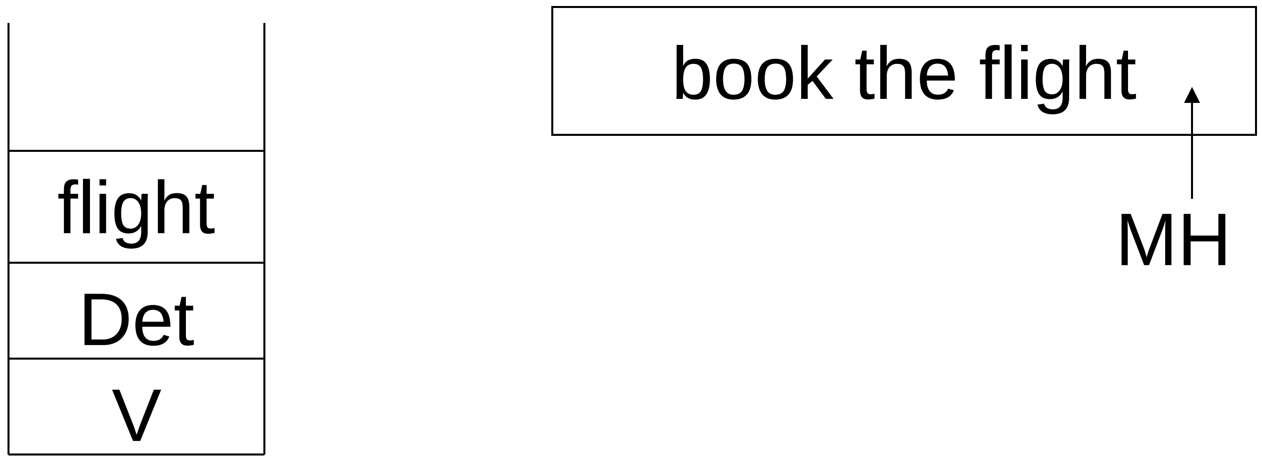
Trace of Bottom-Up Parsing

- Reduce ‘the’



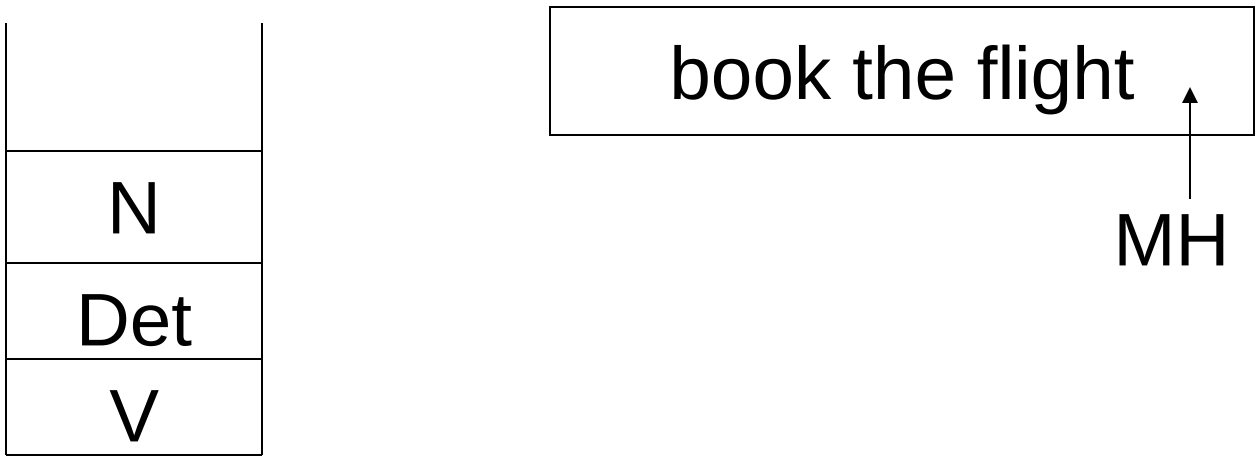
Trace of Bottom-Up Parsing

- Push ‘flight’; advance pointer



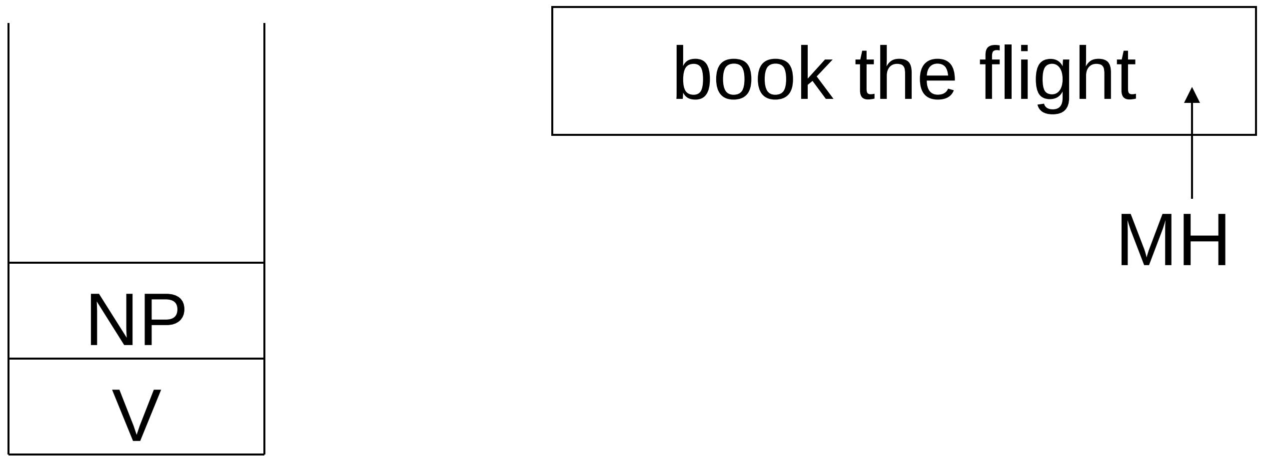
Trace of Bottom-Up Parsing

- Reduce ‘flight’



Trace of Bottom-Up Parsing

- Reduce ‘ART N’ by ‘NP’



Trace of Bottom-Up Parsing

- Reduce ‘V NP’ by ‘S’; *termination* by *S* on stack and input exhausted.