Handout 16: How parsers work

Top-down parser

1. Consider the grammar:

S →	**[a]** NP VP	
NP →	**[a]** Det N N	**[b]** Det N
VP →	**[a]** V	**[b]** V NP

2. Lexicon:

Det →	**[a]** the		
N →	**[a]** police	**[b]** man	**[c]** boat
V →	**[a]** man	**[b]** boat	

3. Parse "the police man the boat"

a. Start from S.

```
[0] * S
```

b. **Expand** category after `*` in every way possible. First way: S → NP VP.

```
[1a] * NP VP
  [2a] * Det N N VP
    [3a] * the N N VP
```

c. Bottomed out. It **matches** the sentence. Advance the `*`.

```
          [4] the * N N VP
```

d. Now back to **expanding**.

```
            [5a] the * police N VP
              [6] the police * N VP
                [7a] the police * police VP
```

e. Does not match. Go to an alternative.

```
                [7b] the police * man VP
                  [8] the police man * VP
                    [9a] the police man * V
                      [10a] the police man * man
                      [10b] the police man * boat
                    [9b] the police man * V NP
                      [11a] the police man * man
                      [11b] the police man * boat
                [7c] the police * boat VP
            [5b] the * man N VP
            [5c] the * boat N VP
      [2b] * Det N VP
```

f. Finally, we're on the right track

```
[12a] * the N VP
  [13] the * N VP
    [14a] the * police VP
      [15] the police * VP
        [16a] the police * V
          [17a] the police * man
            [18] the police man *
```

g. We have generated a complete tree, but the sentence is not done.

```
          [17b] the police * boat
        [16b] the police * V NP
          [19a] the police * man NP
            [20] the police man * NP
              [21a] the police man * Det N N
                [22a] the police man * the N N
                  [23] the police man the * N N
                    [24a] the police man the * police N
                    [24b] the police man the * man N
                    [24c] the police man the * boat N
                      [25] the police man the boat * N
```

h. We consumed all of the input sentence, but we don't have a complete tree.

```
              [21b] the police man * Det N
                [26a] the police man * the N
                  [27] the police man the * N
                    [28a] the police man the * police
                    [28b] the police man the * man
                    [28c] the police man the * boat
```

i. Success! But we continue, to see if there are more parses.

```
            [19b] the police * boat NP
      [14b] the * man VP
      [14c] the * boat VP
```

j. All options exhausted.

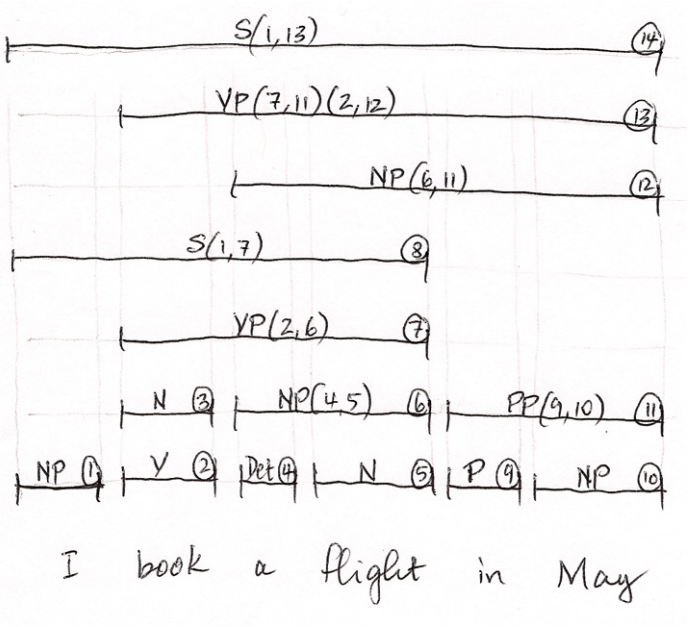
Bottom-up parsing: CKY

4. Grammar

1	S → NP VP	7	NP → I
2	NP → Det N	8	N → book
3	NP → NP PP	9	V → book
4	VP → V NP	10	Det → a
5	VP → VP PP	11	N → flight
6	PP → P NP	12	NP → May

5. Chart

a. The chart



b. Positions:

$_0$ I $_1$ book $_2$ a $_3$ flight $_4$ in $_5$ May $_6$

c. Chart nodes: $_iX_j$. E.g., #1: $_0\text{NP}_1$, #7: $_1\text{VP}_4$.

d. No duplicates: rather, assign multiple child-lists. E.g., #13: $_1\text{VP}_6$.

6. **unwind**(). Read a tree out of the chart: make a non-deterministic choice wherever there is more than one child-list.

```
14          (S
  01          (NP I)
  13          (VP
    07          (VP
      02          (V book)
      06          (NP
        04          (Det a)
        05          (N flight)))
    11          (PP
      09          (P in)
      10          (NP May))))
```

```
14          (S
  01          (NP I)
  13          (VP
    02          (V book)
    12          (NP
      06          (NP
        04          (Det a)
        05          (N flight))
      11          (PP
        09          (P in)
        10          (NP May)))))
```

7. Algorithm for **fill_chart**()
   - **a.** Work from left to right: j ranges from 1 to 6
   - **b.** **shift**(j). Let w be the word ending at position j. For each rule $X \rightarrow w$, create a node ${}_{j-1}X_j$.
   - **c.** **extend_edges**(*node*). Each time you create a node ${}_kZ_j$, look for nodes ${}_iY_k$ ending where ${}_kZ_j$ starts. If there is a rule $X \rightarrow YZ$, then create a new node ${}_iX_j$.
   - **d.** When you create a new node, record the two children that you used to create it.
   - **e.** Do not create duplicates: if you want to create a new node ${}_iX_j$, but it already exists, just add a new child-list to the old edge instead of creating a new one.

8. CKY parser requires grammar in **Chomsky Normal Form.** Only two kinds of rules allowed:

$$X \rightarrow YZ$$

$$X \rightarrow a$$

Bottom-up: chart parser

9. Example: figure 1

10. Two basic data structures

- **a.** Node ${}_iX_j$
- **b.** Edge ${}_iX \to \alpha \bullet_j \beta$

11. Four basic operations (bottom-up parser)

- **a.** **shift**(j). For each part of speech X for word$[j-1]$, create a node ${}_{j-1}X_j$.
- **b.** **bu_predict**(*node*). Invoked after creating node ${}_iY_j$. For each rule $X \to Y\beta$, create edge ${}_iX \to Y \bullet_j \beta$.
- **c.** **extend_edges**(*node*). Invoked after creating node ${}_kZ_j$. For each edge ${}_iX \to \alpha \bullet_k Z\beta$, create edge ${}_iX \to \alpha Z \bullet_j \beta$.
- **d.** **complete**(*edge*). Invoked after creating edge ${}_iX \to \alpha \bullet_j$. Create node ${}_iX_j$. But if ${}_iX_j$ already exists, just add a new expansion instead.

12. Two supporting operations that stitch everything together:

- **a.** **create_node**(i, X, j). Create node ${}_iX_j$ and put it in the chart. Then call `bu_predict` and `extend_edges`.
- **b.** **create_edge**(i, X, α, j, β). Create the edge ${}_iX \to \alpha \bullet_j \beta$ and put it in the chart. If the dot is at the end, call `complete`.

13. Top level

- **a.** **fill_chart**(*words*). Initialize the chart, then call `shift` at each sentence position, from 1 to the length of the sentence. Each call to `shift` triggers a cascade of actions.
- **b.** **unwind**(). Look for an S node spanning the whole sentence. If found, extract the list of trees from it and return them.

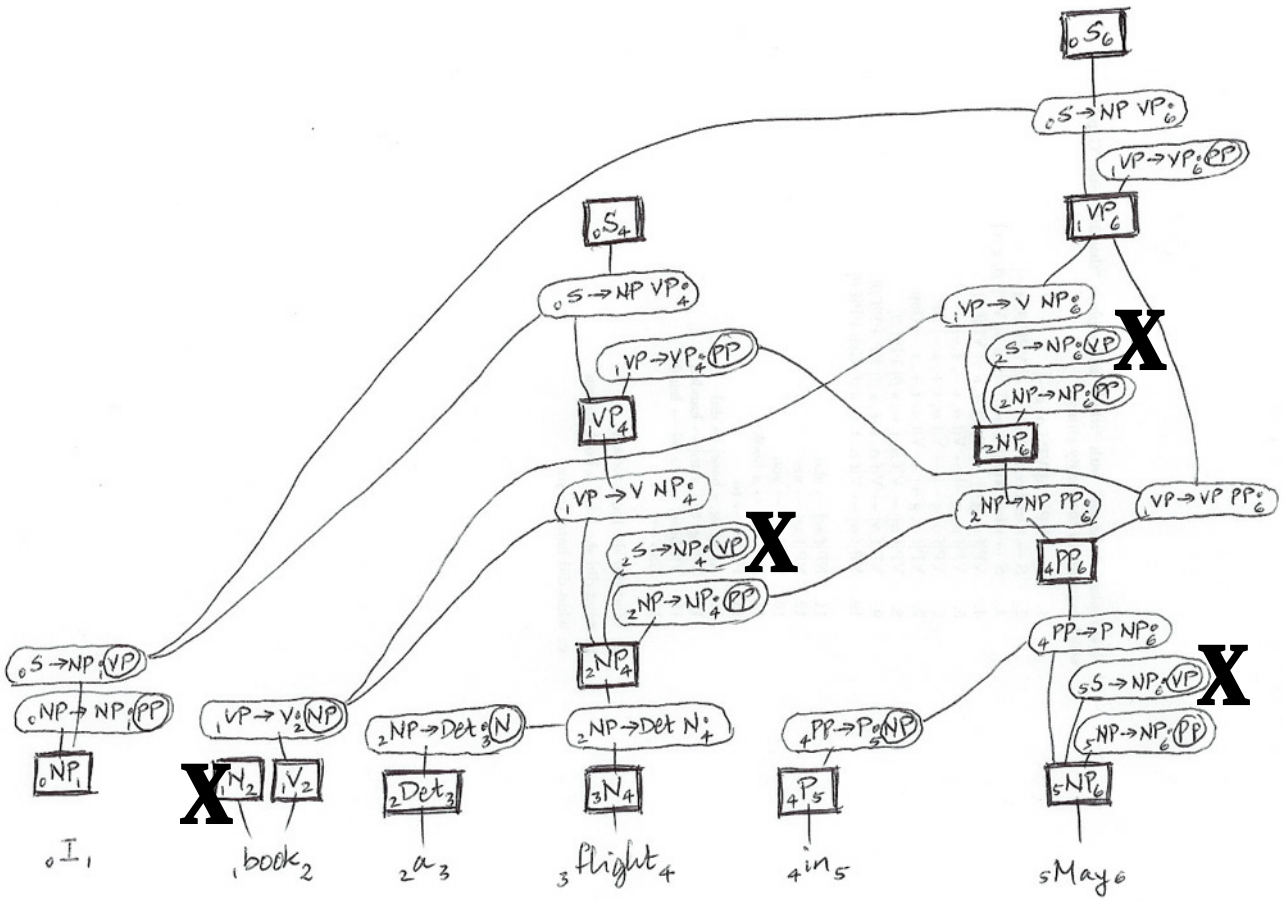


Figure 1: Filled chart for sentence "I book a flight in May." The node and edges marked with "X" are filtered out when topdown prediction is used. (Not shown is an initial prediction $\rightarrow \bullet_0 S$.)