Natural Language Processing
SoSe 2015

# Parsing

*Dr. Mariana Neves*

*May 18th, 2014*

(based on the slides of Dr. Saeedeh Momtazi)

# Parsing

- Finding structural relationships between words in a sentence

```
(ROOT
  (S
    (NP
      (NP (NNP Steve) (NNP Paul) (NNP Jobs))
      (, ,)
      (NP
        (NP (NN co-founder))
        (PP (IN of)
          (NP (NNP Apple) (NNP Inc))))
      (, ,))
    (VP (VBD was)
      (VP (VBN born)
        (PP (IN in)
          (NP (NNP California)))))
    (. .)))
```

(http://nlp.stanford.edu:8080/parser)

# Parsing

- Applications
  - Grammar checking
  - Speech recognition
  - Machine translation
  - Relation extraction
  - Question answering

(http://nlp.stanford.edu:8080/parser)

# Parsing

- Grammar checking
  - By failing to parse a sentence

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBP buy)
      (NP (DT the) (NN book)))))
```

(http://nlp.stanford.edu:8080/parser)

# Parsing

- Speech recognition
  - By failing to parse a sentence

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBP buy)
      (NP (DT a) (NN book)))))
```

```
(ROOT
  (FRAG
    (NP (NNP John))
    (PP (IN by)
      (NP (DT a) (NN book)))))
```

(http://nlp.stanford.edu:8080/parser)

# Parsing

- Machine translation
  - Fail to parse a sentence

**Babel interaktiv: „Ich mag Brot*"**

Analyse für den Satz „Ich mag Brot*"

```
             v(fin, ind)
      ___________|___________
     /           |           \
n(nom, sg)  v(fin, ind)  n(acc, sg)
     |           |           |
     |           |           |
n(nom, sg)      mag      n(acc, sg)
     |                       |
     |                       |
    ich                     brot
```

**Babel interaktiv: „Ich wie Brot*"**

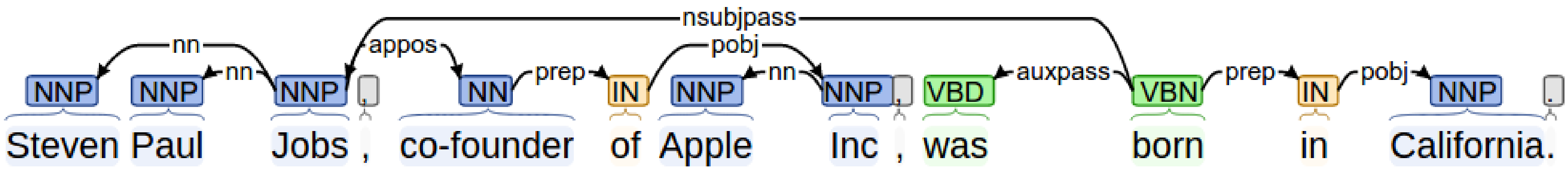
Analyse für den Satz „Ich wie Brot*"

Keine Analyse gefunden! Warum?

(http://hpsg.fu-berlin.de/~stefan/cgi-bin/babel.cgi)

# Parsing

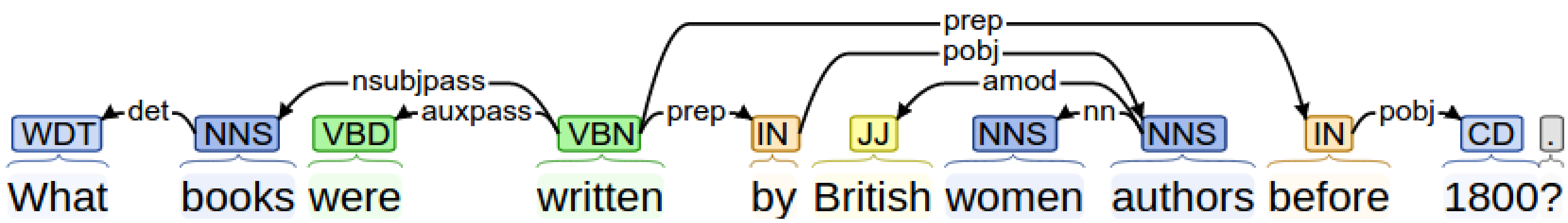
- Relation extraction




(http://nlp.stanford.edu:8080/corenlp/)

# Parsing

- Question answering




(http://nlp.stanford.edu:8080/corenlp/)

# Outline

- Phrase Structure
- Syntactic Parsing
  - CKY Algorithm
- Statistical Parsing

# Outline

- Phrase Structure

- Syntactic Parsing
  - CKY Algorithm

- Statistical Parsing

# Constituency

- Working based on Constituency (Phrase structure)
  - Organizing words into nested constituents

```
(ROOT
  (S
    (NP (PRP$ My) (NN dog))
    (ADVP (RB also))
    (VP (VBZ likes)
      (S
        (VP (VBG eating)
          (NP (NN sausage)))))
    (. .)))
```

(http://nlp.stanford.edu:8080/parser)

# Constituency

- Working based on Constituency (Phrase structure)
  - Showing that groups of words can act as single units

```
(ROOT
  (S
    (NP (PRP$ My) (NN dog))
    (ADVP (RB also))
    (VP (VBZ likes)
      (S
        (VP (VBG eating)
          (NP (NN sausage)))))
    (. .)))
```

(http://nlp.stanford.edu:8080/parser)

# Constituency

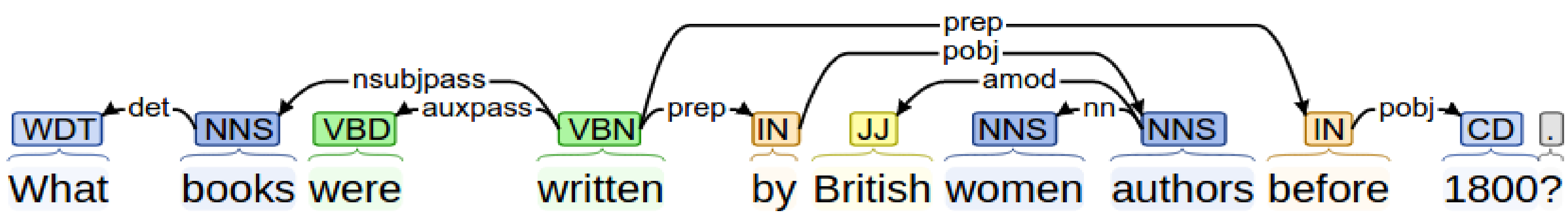
- Working based on Constituency (Phrase structure)
  - Forming coherent classes from these units that can behave in similar ways
    - With respect to their internal structure
    - With respect to other units in the language

```
(ROOT
  (S
    (NP (PRP$ My) (NN dog))
    (ADVP (RB also))
    (VP (VBZ likes)
      (S
        (VP (VBG eating)
          (NP (NN sausage)))))
    (. .)))
```

(http://nlp.stanford.edu:8080/parser)

# Constituency

- Working based on Constituency (Phrase structure)
  - Considering a head word for each constituent




```
(ROOT
  (SBARQ
    (WHNP (WP What))
    (SQ
      (NP (NNS books))
      (VP (VBD were)
        (VP (VBN written)
          (PP (IN by)
            (NP (JJ British) (NNS women) (NNS authors)))
          (PP (IN before)
            (NP (CD 1800))))))
    (. ?)))
```

(http://nlp.stanford.edu:8080/parser)

# Constituency

The writer talked to the audience about his new book.

The writer talked about his new book to the audience.

About his new book the writer talked to the audience.

The writer talked about to the audience his new book.

# Constituency

The writer talked to the audience about his new book.

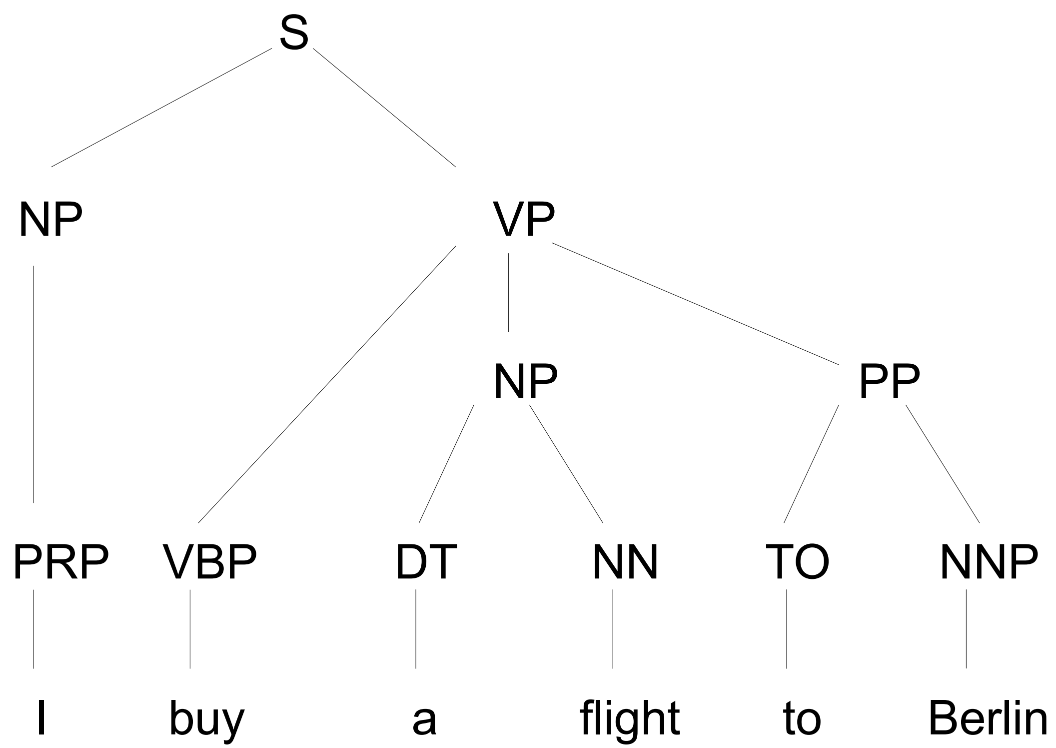
The writer talked about his new book to the audience. ✓

About his new book the writer talked to the audience. ✓

The writer talked about to the audience his new book. ✗

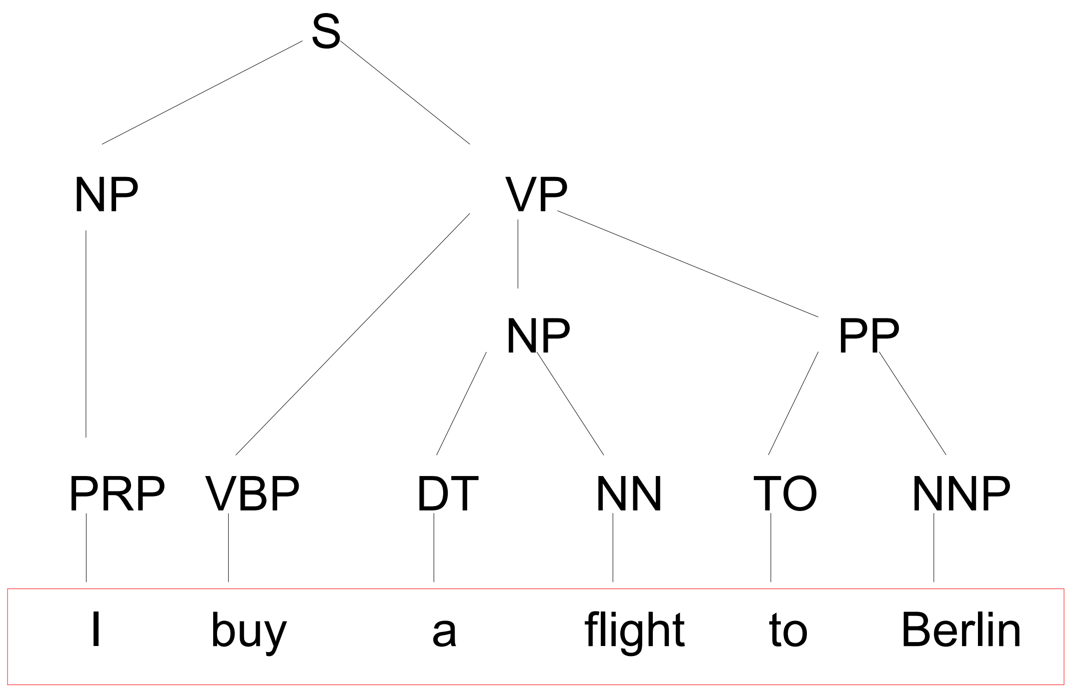
# Context Free Grammar (CFG)

- Grammar G consists of
  - Terminals (T )
  - Non-terminals (N)
  - Start symbol (S)
  - Rules (R)



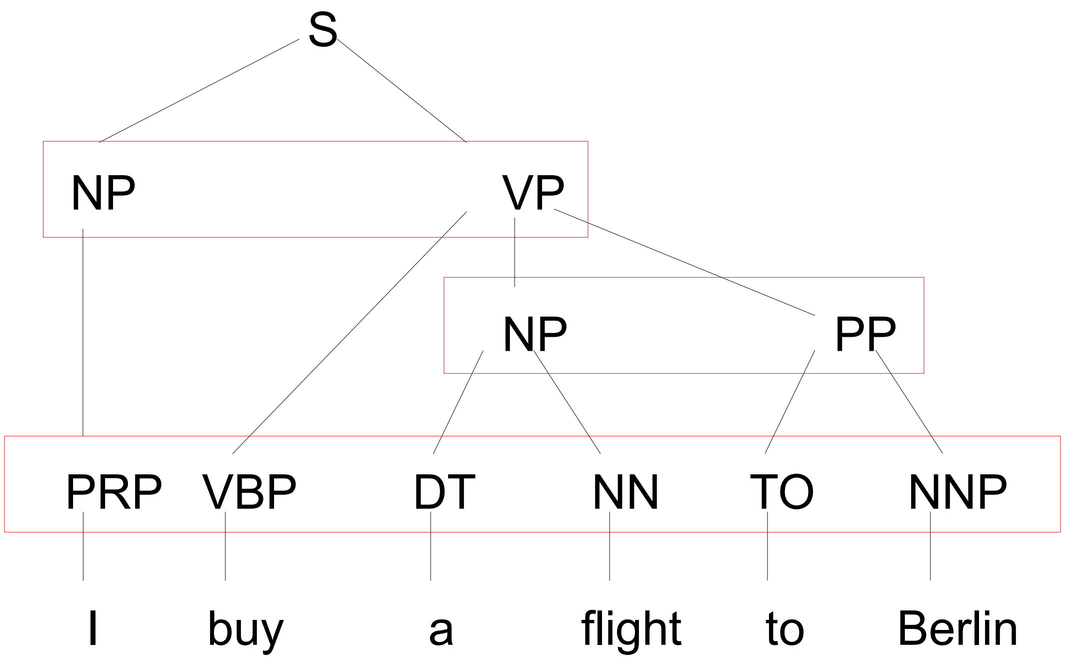
# Context Free Grammar (CFG)

- Terminals
  - The set of words in the text



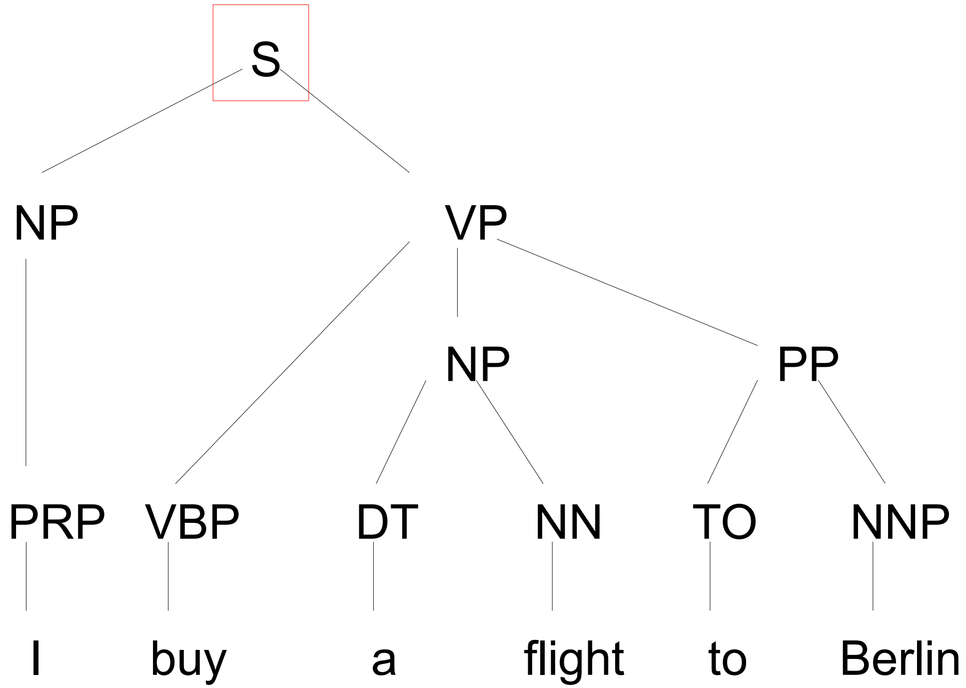
# Context Free Grammar (CFG)

- Non-Terminals
  - The constituents in a language (noun phrase, verb phrase, ....)



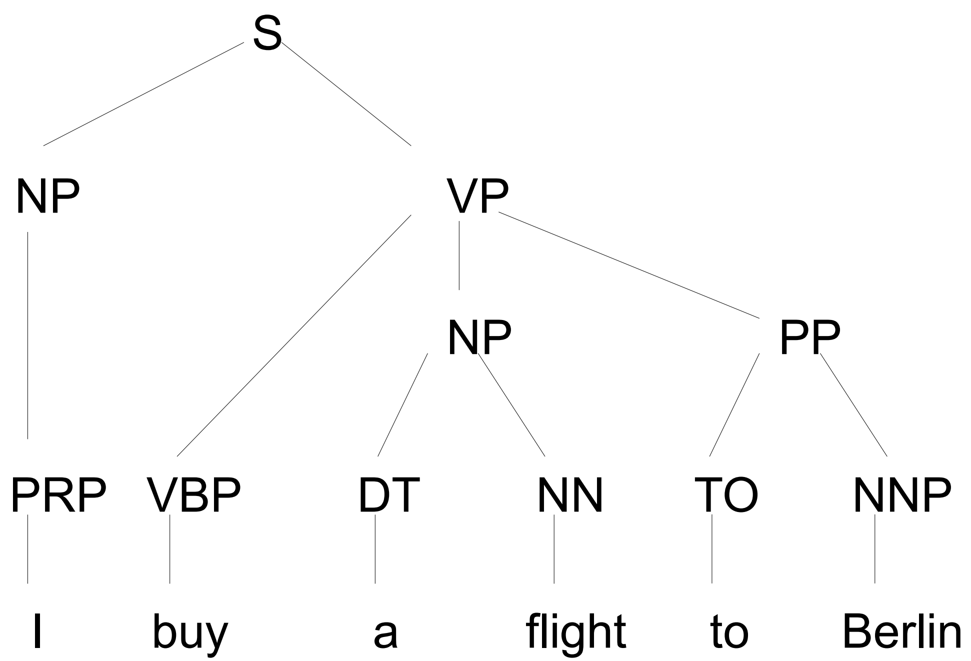
# Context Free Grammar (CFG)

- Start symbol
  - The main constituent of the language (sentence)



# Context Free Grammar (CFG)

- Rules
  - Equations that consist of a single non-terminal on the left and any number of terminals and non-terminals on the right




$S \rightarrow NP\ VP$

# Context Free Grammar (CFG)

S → NP VP

S → VP

NP → NN

NP → PRP

NP → DT NN

NP → NP NP

NP → NP PP

VP → VBP NP

VP → VBP NP PP

VP → VP PP

VP → VP NP

PP → TO NNP

PRP → I

NN → book

VBP → buy

DT → a

NN → flight

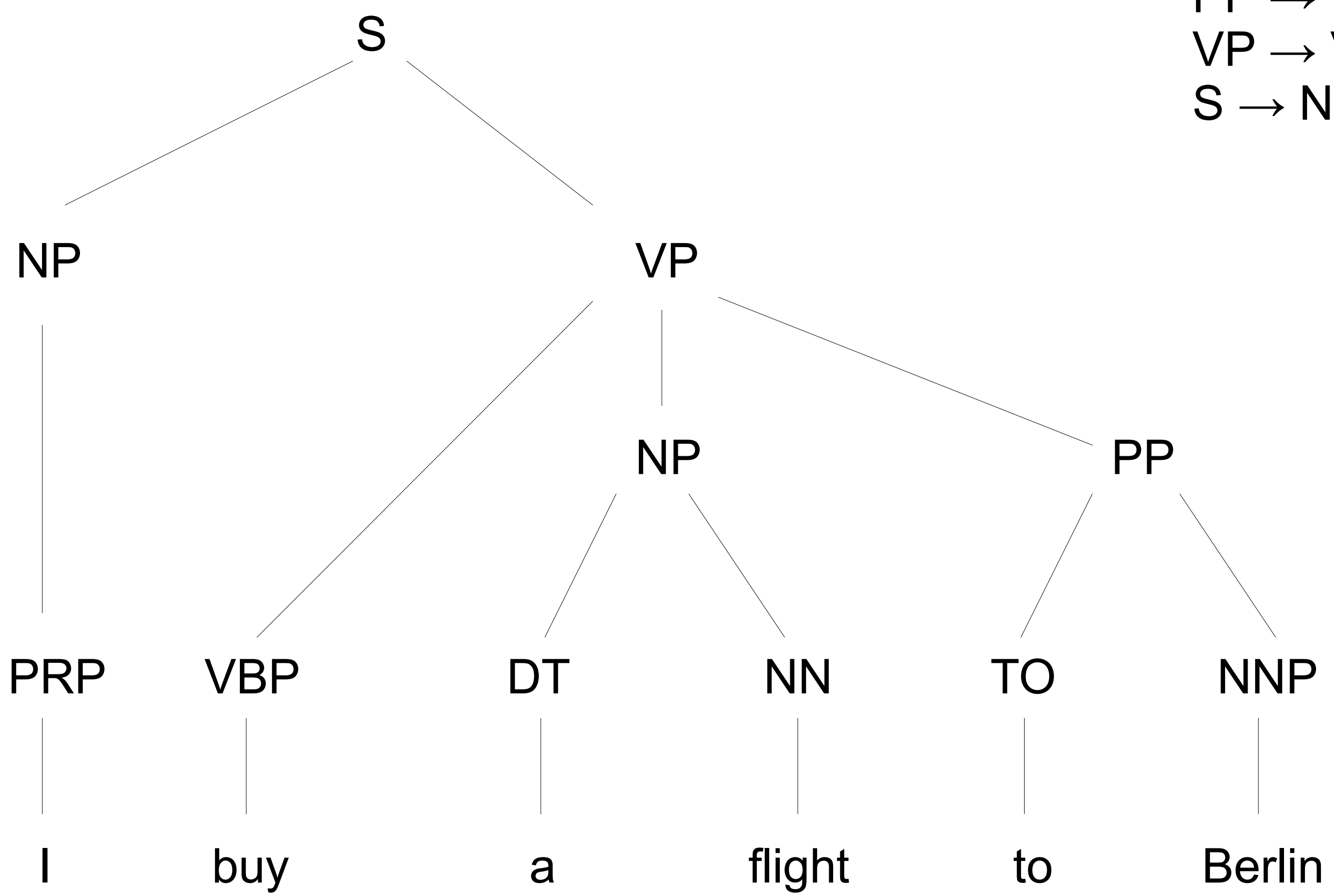
TO → to

NNP → Berlin

# CFG

# CFG

$NP \rightarrow PRP$
$NP \rightarrow DT\ NN$
$PP \rightarrow TO\ NNP$
$VP \rightarrow VBP\ NP\ PP$
$S \rightarrow NP\ VP$



# Outline

- Phrase Structure
- Syntactic Parsing
  - CKY Algorithm
- Statistical Parsing

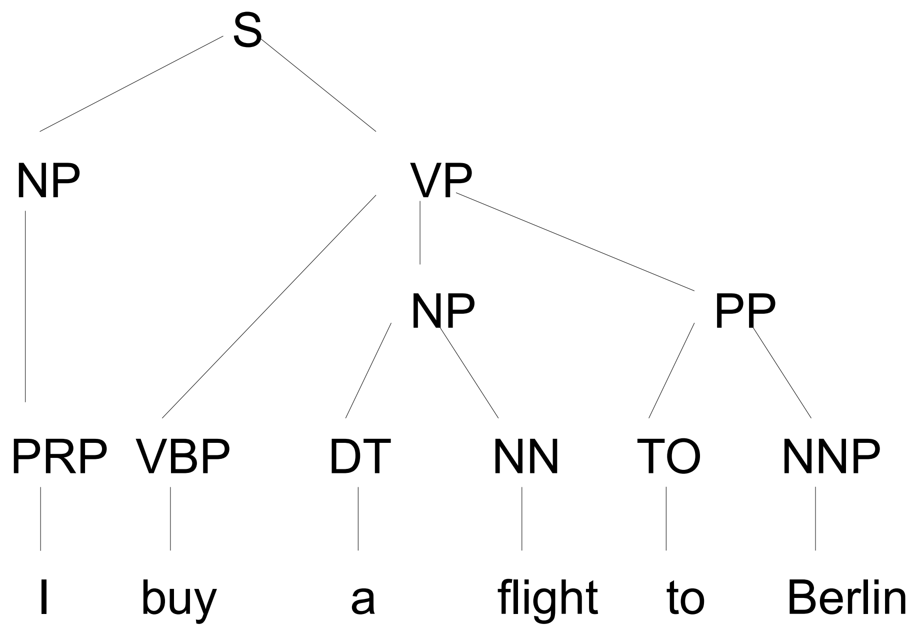
# Parsing

- Taking a string and a grammar and returning proper parse tree(s) for that string

NP → PRP
NP → DT NN
PP → TO NNP
VP → VBP NP PP
S → NP VP

+

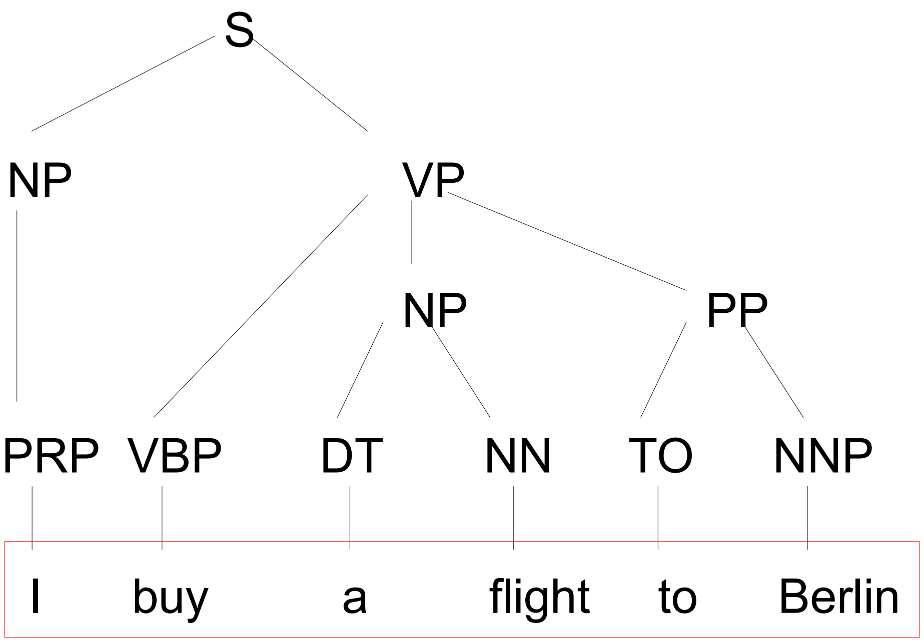
I buy a flight to Berlin.



# Parsing

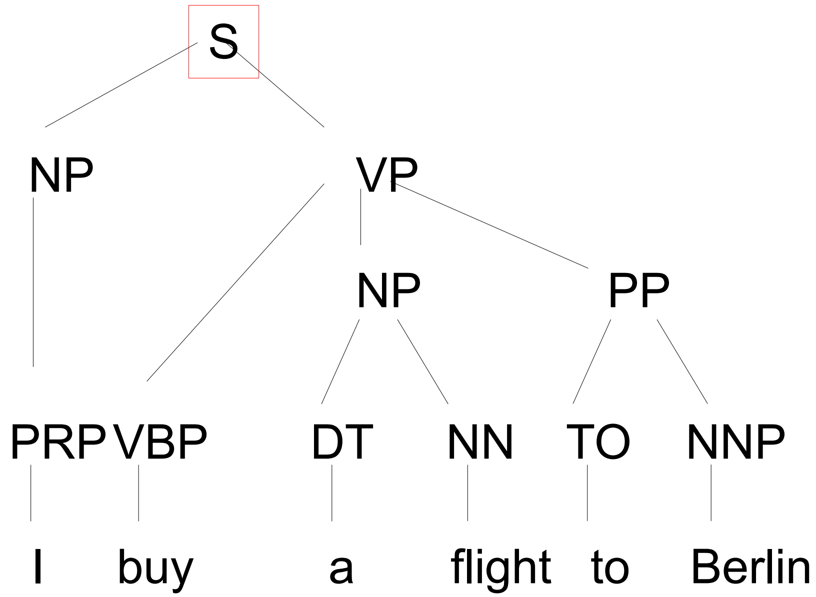
- Covering all and only the elements of the input string

I buy a flight to Berlin.



# Parsing

- Reaching the start symbol at the top of the string



# Main Grammar Fragments

- Sentence
- Noun Phrase
  - Agreement
- Verb Phrase
  - Sub-categorization

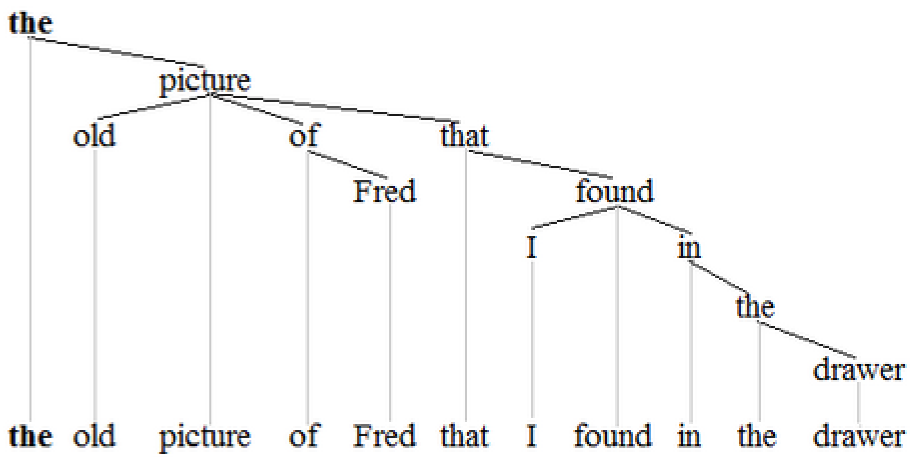
# Grammar Fragments: Sentence

- Declaratives
  - A plane left.
  - S → NP VP
- Imperatives
  - Leave!
  - S → VP
- Yes-No Questions
  - Did the plane leave?
  - S → Aux NP VP
- Wh Questions
  - Which airlines fly from Berlin to London?
  - S → Wh-NP VP

# Grammar Fragments: NP

- Each NP has a central critical noun called head

- The head of an NP can be expressed using
  - Pre-nominals: the words that can come before the head
  - Post-nominals: the words that can come after the head




(http://en.wikipedia.org/wiki/Noun_phrase)

# Grammar Fragments: NP

- Pre-nominals
  - Simple lexical items: the, this, a, an, ...
    - a car
  - Simple possessives
    - John's car
  - Complex recursive possessives
    - John's sister's friend's car
  - Quantifiers, cardinals, ordinals...
    - three cars
  - Adjectives
    - large cars

# Grammar Fragments: NP

- Post-nominals
  - Prepositional phrases
    - I book a flight from Seattle
  - Non-finite clauses (-ing, -ed, infinitive)
    - There is a flight arriving before noon
    - I need to have dinner served
    - Which is the last flight to arrive in Boston?
  - Relative clauses
    - I want a flight that serves breakfast

# Agreement

- Having constraints that hold among various constituents
- Considering these constraints in a rule or set of rules
- Example: determiners and the head nouns in NPs have to agree in number
  - This flight
  - Those flights
  - This flights
  - Those flight

# Agreement

- Grammars that do not consider constraints will over-generate
  - Accepting and assigning correct structures to grammatical examples (this flight)
  - But also accepting incorrect examples (these flight)

# Agreement at sentence level

- Considering similar constraints at sentence level
- Example: subject and verb in sentences have to agree in number and person

- John flies
- We fly
- John fly
- We flies

# Agreement

- How to solve the agreement problem in parsing?

  - This flight
  - Those flights
  - This flights
  - Those flight

- John flies
- We fly
- John fly
- We flies

# Agreement

- Possible CFG solution
  - $S_{sg} \rightarrow NP_{sg}\ VP_{sg}$
  - $S_{pl} \rightarrow NP_{pl}\ VP_{pl}$
  - $NP_{sg} \rightarrow Det_{sg}\ N_{sg}$
  - $NP_{pl} \rightarrow Det_{pl}\ N_{pl}$
  - $VP_{sg} \rightarrow V_{sg}\ NP_{sg}$
  - $VP_{pl} \rightarrow V_{pl}\ NP_{pl}$
  - ...
- Shortcoming:
  - Introducing many rules in the system

# Grammar Fragments: VP

- VPs consist of a head verb along with zero or more constituents called arguments
  - VP → V (disappear)
  - VP → V NP (prefer a morning flight)
  - VP → V PP (fly on Thursday)
  - VP → V NP PP (leave Boston in the morning)
  - VP → V NP NP (give me the flight number)
- Arguments
  - Obligatory: complement
  - Optional: adjunct

# Grammar Fragments: VP

- Even though there are many valid VP rules, not all verbs are allowed to participate in all VP rules
  - disappear a morning flight

# Grammar Fragments: VP

- Solution (Sub-categorization):
  - Sub-categorizing the verbs according to the sets of VP rules that they can participate in
  - Modern grammars have more than 100 subcategories

# Sub-categorization

- Example:
  - sneeze: John sneezed
  - find: Please find [a flight to NY]$_{NP}$
  - give: Give [me]$_{NP}$ [a cheaper fair]$_{NP}$
  - help: Can you help [me]$_{NP}$ [with a flight]$_{PP}$
  - prefer: I prefer [to leave earlier]$_{TO\text{-}VP}$
  - tell: I was told [United has a flight]$_{S}$

- John sneezed the book
- I prefer United has a flight
- Give with a flight

# Sub-categorization

- The over-generation problem also exists in VP rules
  - Permitting the presence of strings containing verbs and arguments that do not go together
  - John sneezed the book
    - VP → V NP
- Solution:
  - Similar to agreement phenomena, we need a way to formally express the constraints

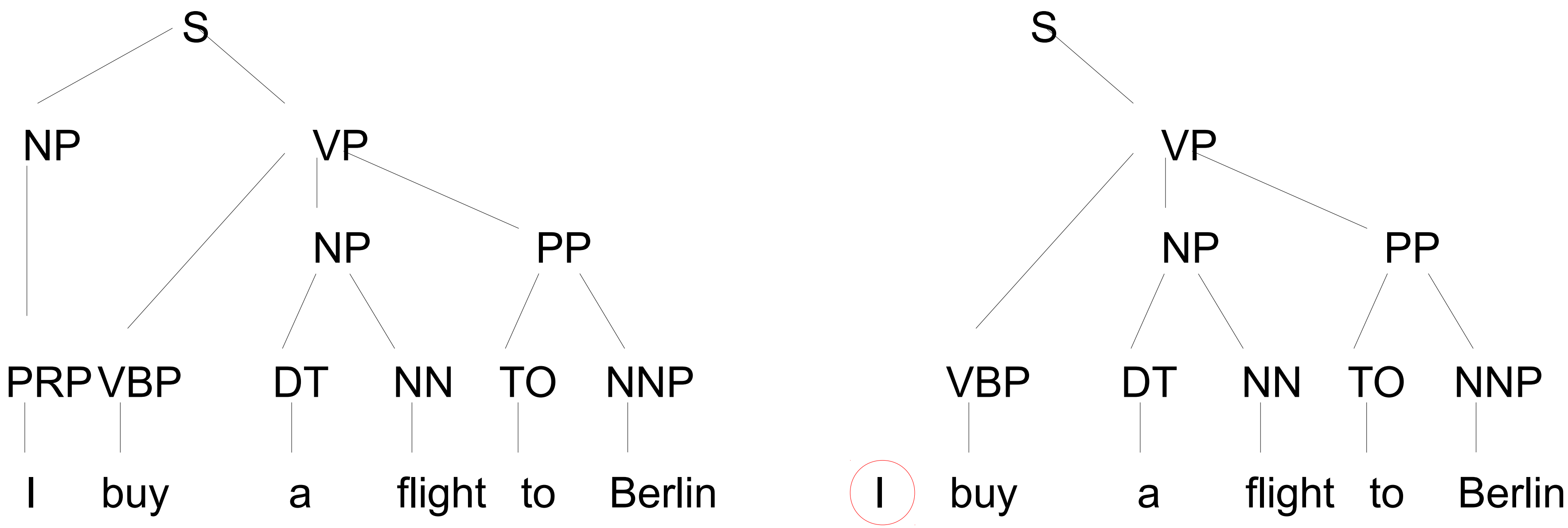
# Parsing Algorithms

- Top-Down
- Bottom-up

# Parsing Algorithms

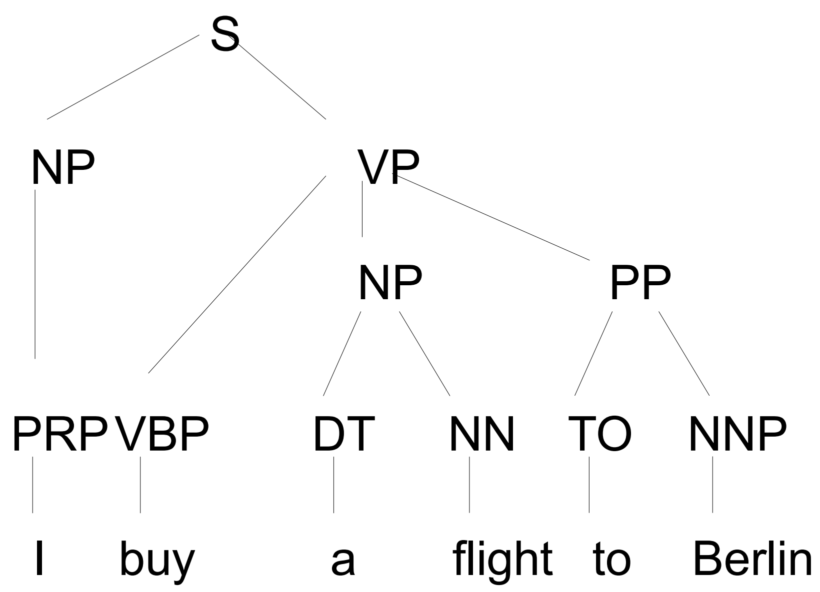
- Top-Down
  - Starting with the rules that give us an S
  - Working on the way down from S to the words

S → NP VP
S → VP



# Parsing Algorithms

- Bottom-Up
  - Starting with trees that link up with the words
  - Working on the way up from words to larger and larger trees



# Top-Down vs. Bottom-Up

- Advantages

- Disadvantages

# Top-Down vs. 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 suggests trees that make no sense globally

# Top-Down vs. Bottom-Up

- In both cases;
  - keep track of the search space and make choices
- Solutions
  - Backtracking
    - Making a choice, if it works out then fine
    - If not, then back up and make a different choice ⇒ duplicated work
  - Dynamic programming
    - Avoiding repeated work
    - Solving exponential problems in polynomial time
    - Storing ambiguous structures efficiently

# Dynamic Programming Methods

- CKY (Cocke-Kasami-Younger): bottom-up
- Early: top-down

# Outline

- Phrase Structure
- Syntactic Parsing
  - CKY Algorithm
- Statistical Parsing

# Chomsky Normal Form (CNF)

- Each grammar can be represented by a set of binary rules
  - $A \rightarrow B\ C$
  - $A \rightarrow w$

- A, B, C are non-terminals; w is a terminal

# Chomsky Normal Form

- Converting to Chomsky Normal Form

$$A \rightarrow B\ C\ D$$

$$X \rightarrow B\ C$$

$$A \rightarrow X\ D$$

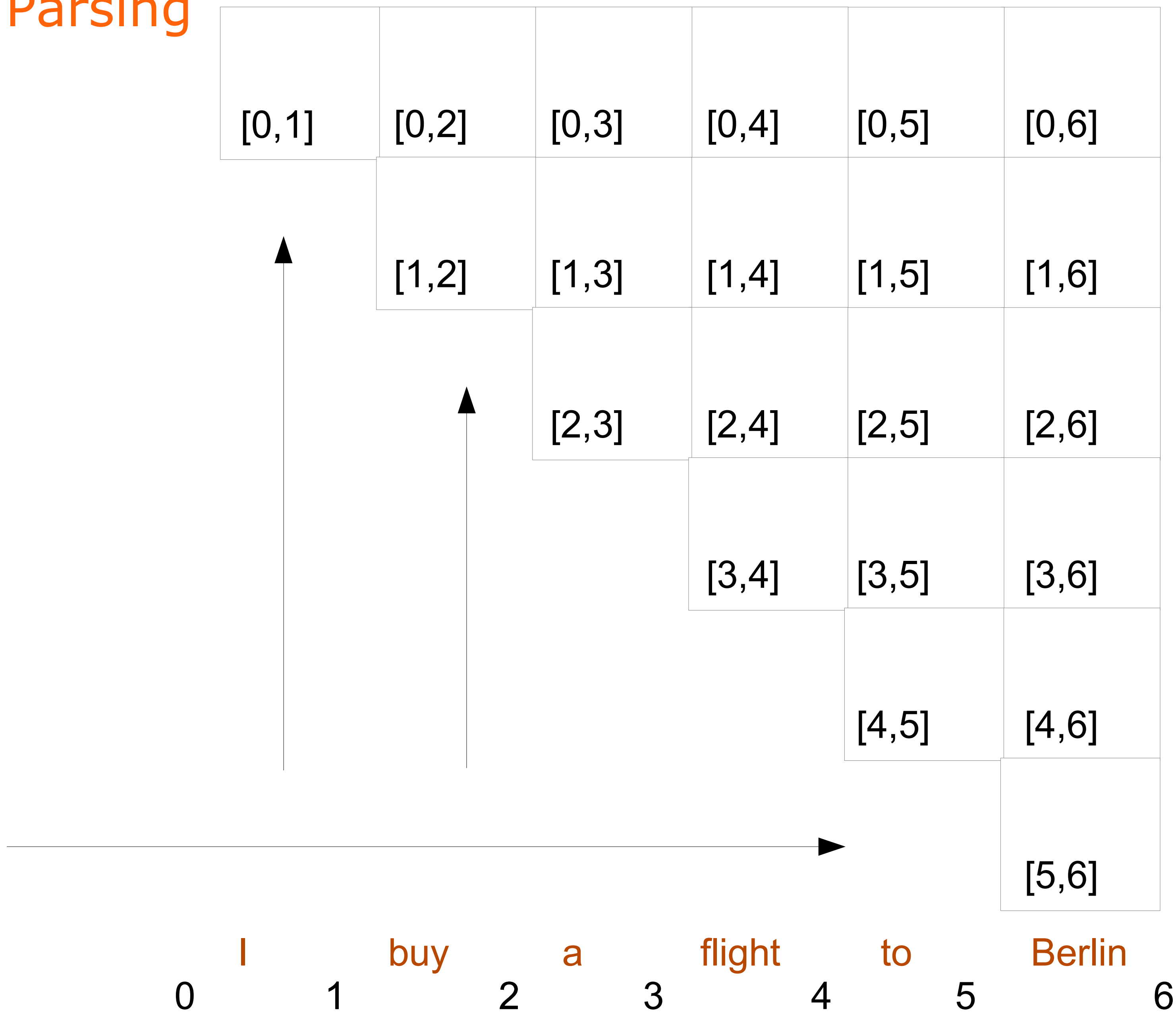
# CKY Parsing

$$A \rightarrow B\ C$$

- If there is an A somewhere in the input, then there must be a B followed by a C in the input
- If the A spans from *i* to *j* in the input, then there must be a *k* such that $i < k < j$
  - B spans from *i* to *k*
  - C spans from *k* to *j*

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | I | | buy | | a | | flight | | to | | Berlin | |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |
| i | | k | | | | | | | | | | j |

# CKY Parsing


[0,1]
[0,2]
[0,3]
[0,4]
[0,5]
[0,6]
[1,2]
[1,3]
[1,4]
[1,5]
[1,6]
[2,3]
[2,4]
[2,5]
[2,6]
[3,4]
[3,5]
[3,6]
[4,5]
[4,6]
[5,6]
I
buy
a
flight
to
Berlin
0
1
2
3
4
5
6

# CKY Parsing

PRP → I
NP → PRP

| | | | | | |
|---|---|---|---|---|---|
| PRP, NP [0,1] | [0,2] | [0,3] | [0,4] | [0,5] | [0,6] |
| | [1,2] | [1,3] | [1,4] | [1,5] | [1,6] |
| | | [2,3] | [2,4] | [2,5] | [2,6] |
| | | | [3,4] | [3,5] | [3,6] |
| | | | | [4,5] | [4,6] |
| | | | | | [5,6] |

I buy a flight to Berlin

0 1 2 3 4 5 6

# CKY Parsing

PRP → I
NP → PRP

VBP → buy

| | | | | | |
|---|---|---|---|---|---|
| PRP, NP [0,1] | [0,2] | [0,3] | [0,4] | [0,5] | [0,6] |
| | VBP [1,2] | [1,3] | [1,4] | [1,5] | [1,6] |
| | | [2,3] | [2,4] | [2,5] | [2,6] |
| | | | [3,4] | [3,5] | [3,6] |
| | | | | [4,5] | [4,6] |
| | | | | | [5,6] |

I buy a flight to Berlin

0 1 2 3 4 5 6

# CKY Parsing

PRP → I
NP → PRP

VBP → buy

DT → a

| | | | | | |
|---|---|---|---|---|---|
| PRP, NP [0,1] | [0,2] | [0,3] | [0,4] | [0,5] | [0,6] |
| | VBP [1,2] | [1,3] | [1,4] | [1,5] | [1,6] |
| | | DT [2,3] | [2,4] | [2,5] | [2,6] |
| | | | [3,4] | [3,5] | [3,6] |
| | | | | [4,5] | [4,6] |
| | | | | | [5,6] |

I buy a flight to Berlin

0 1 2 3 4 5 6

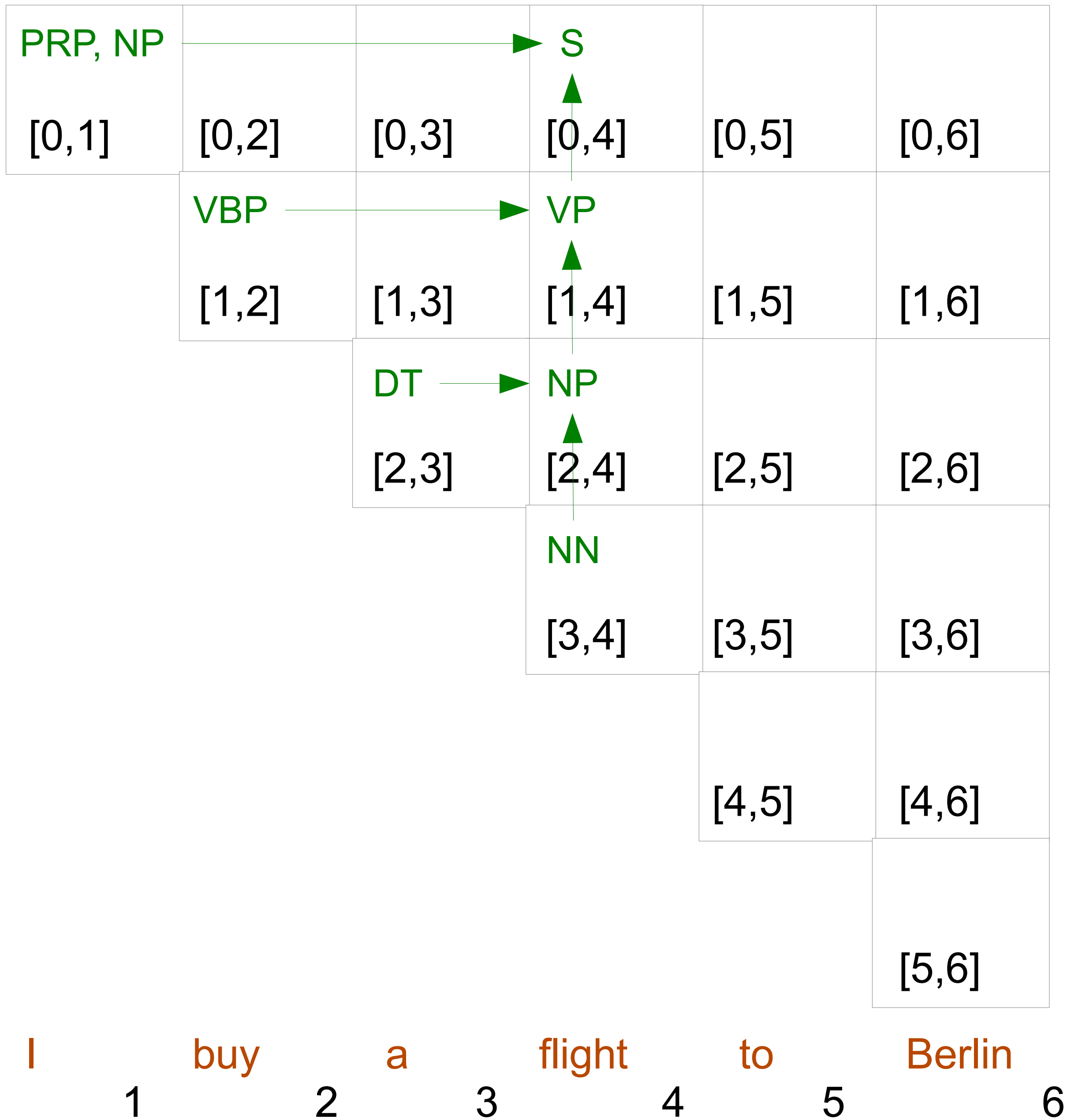
# CKY Parsing

PRP → I
NP → PRP

VBP → buy

DT → a

NN → flight
NP → DT NN
VP → VBP NP
S → NP VP



# CKY Parsing

PRP → I
NP → PRP

VBP → buy

DT → a

NN → flight
NP → DT NN
VP → VBP NP
S → NP VP

TO → to

| | | | | | |
|---|---|---|---|---|---|
| PRP, NP [0,1] | [0,2] | [0,3] | S [0,4] | [0,5] | [0,6] |
| | VBP [1,2] | [1,3] | VP [1,4] | [1,5] | [1,6] |
| | | DT [2,3] | NP [2,4] | [2,5] | [2,6] |
| | | | NN [3,4] | [3,5] | [3,6] |
| | | | | TO [4,5] | [4,6] |
| | | | | | [5,6] |

I buy a flight to Berlin

0 1 2 3 4 5 6

# CKY Parsing

PRP → I
NP → PRP

VBP → buy

DT → a

NN → flight
NP → DT NN
VP → VBP NP
S → NP VP

TO → to

NNP → Berlin
PP → TO NNP
VP → VP PP

| | | | | | |
|---|---|---|---|---|---|
| PRP, NP [0,1] | [0,2] | [0,3] | S [0,4] | [0,5] | S [0,6] |
| | VBP [1,2] | [1,3] | VP [1,4] | [1,5] | VP [1,6] |
| | | DT [2,3] | NP [2,4] | [2,5] | [2,6] |
| | | | NN [3,4] | [3,5] | [3,6] |
| | | | | TO [4,5] | PP [4,6] |
| | | | | | NNP [5,6] |

I buy a flight to Berlin

0 1 2 3 4 5 6

# Outline

- Phrase Structure
- Syntactic Parsing
  - CKY Algorithm
- Statistical Parsing

# Probabilistic Context Free Grammar (PCFG)

- Terminals (T )
- Non-terminals (N)
- Start symbol (S)
- Rules (R)
- Probability function (P)

Hasso Plattner Institut

# Context Free Grammar (CFG)

S → NP VP

S → VP

NP → NN

NP → PRP

NP → DT NN

NP → NP NP

NP → NP PP

VP → VBP NP

VP → VP PP

VP → VP NP

PP → TO NNP

PRP → I

NN → book

VBP → buy

DT → a

NN → flight

TO → to

NNP → Berlin

# Probabilistic Context Free Grammar

| | |
|---|---|
| 0.9 | S → NP VP |
| 0.1 | S → VP |
| 0.3 | NP → NN |
| 0.4 | NP → PRP |
| 0.1 | NP → DT NN |
| 0.2 | NP → NP NP |
| 0.1 | NP → NP PP |
| 0.4 | VP → VBP NP |
| 0.3 | VP → VP PP |
| 0.5 | VP → VP NP |
| 1.0 | PP → TO NNP |

| | |
|---|---|
| 1.0 | PRP → I |
| 0.6 | NN → book |
| 0.7 | VBP → buy |
| 0.8 | DT → a |
| 0.4 | NN → flight |
| 1.0 | TO → to |
| 1.0 | NNP → Berlin |

# Treebank

- A treebank is a corpus in which each sentence has been paired with a parse tree
- These are generally created by
  - Parsing the collection with an automatic parser
  - Correcting each parse by human annotators if required

```
(S ▷(NP ▷(NP ▷Mice◁) (ADJP ▷transgenic (PP ▷for (NP ▷the (NP ▷(NP ▷human T
cell leukemia virus◁) (PRN ▷((NP ▷HTLV-I◁))◁)◁) Tax gene◁)◁)◁)◁) (VP ▷develop
(NP ▷(NP ▷fibroblastic tumors◁) (SBAR ▷(WHNP ▷that◁) (S ▷(NP ▷◁)(VP ▷express
(NP ▷(ADJP ▷NF-kappa B-inducible◁) early genes◁)◁)◁)◁)◁)◁).◁)
```

(http://www.nactem.ac.uk/aNT/genia.html)

# Penn Treebank

- Penn Treebank is a widely used treebank for English
  - Most well-known section: Wall Street Journal Section
  - 1 M words from 1987-1989

```
(S (NP (NNP John))
   (VP (VPZ flies)
       (PP (IN to)
           (NNP Paris)))
   (. .))
```

# Statistical Parsing

- Considering the corresponding probabilities while parsing a sentence
- Selecting the parse tree which has the highest probability

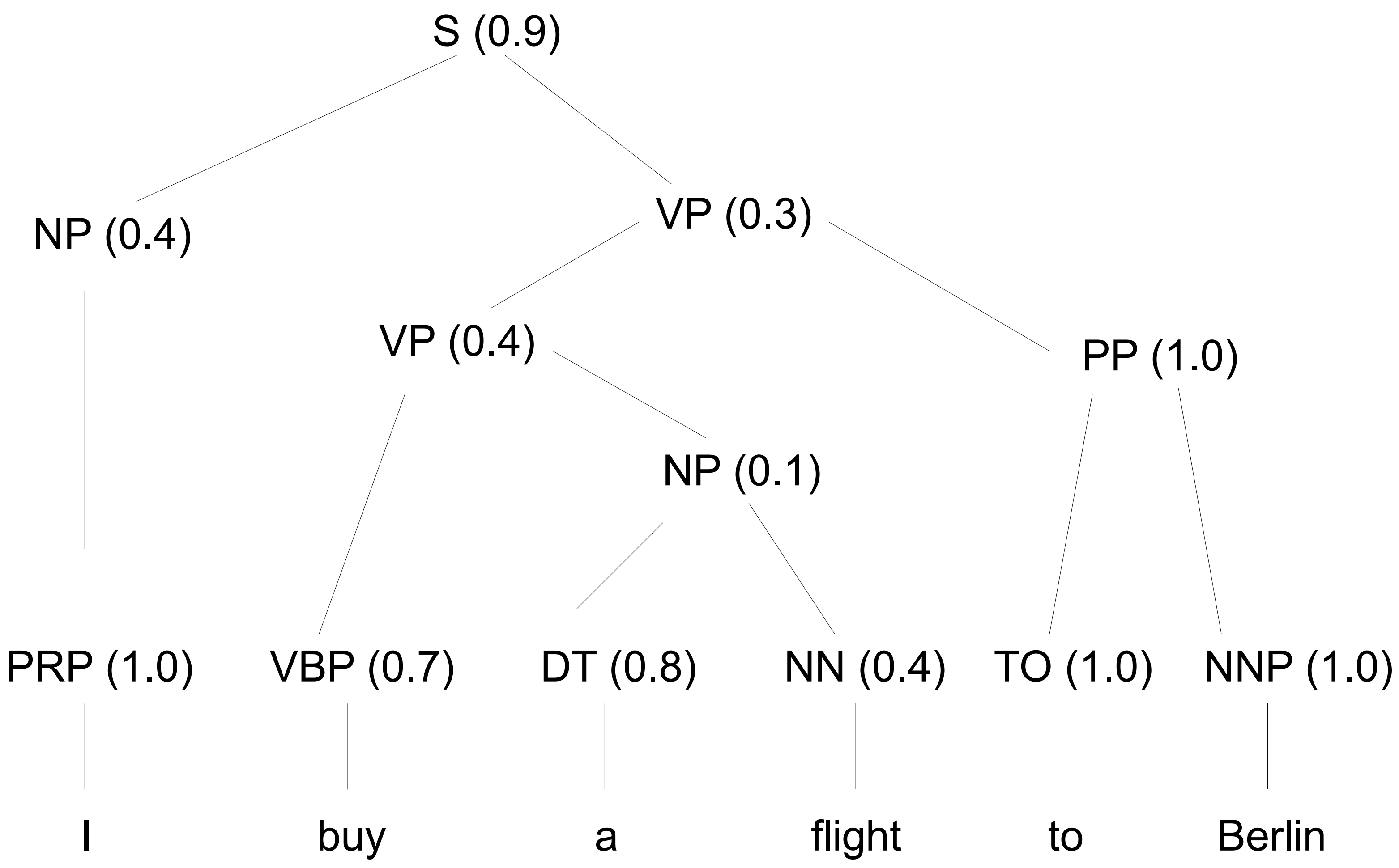
- P(t): the probability of a tree t
  - Product of the probabilities of the rules used to generate the tree

# Probabilistic Context Free Grammar

| | |
|---|---|
| 0.9 | S → NP VP |
| 0.1 | S → VP |
| 0.3 | NP → NN |
| 0.4 | NP → PRP |
| 0.1 | NP → DT NN |
| 0.2 | NP → NP NP |
| 0.1 | NP → NP PP |
| 0.4 | VP → VBP NP |
| 0.3 | VP → VP PP |
| 0.5 | VP → VP NP |
| 1.0 | PP → TO NNP |

| | |
|---|---|
| 1.0 | PRP → I |
| 0.6 | NN → book |
| 0.7 | VBP → buy |
| 0.8 | DT → a |
| 0.4 | NN → flight |
| 1.0 | TO → to |
| 1.0 | NNP → Berlin |

# Statistical Parsing




$$P(t) = 0.9 \times 0.4 \times 1.0 \times 0.3 \times 0.4 \times 0.7 \times 0.1 \times 0.8 \times 0.4 \times 1.0 \times 1.0 \times 1.0$$

# Probabilistic CKY Parsing

PRP → I (1.0)
NP → PRP (0.4)

VBP → buy (0.7)

DT → a (0.8)

NN → flight (0.4)
NP → DT NN (0.1)
VP → VBP NP (0.4)
S → NP VP (0.9)

TO → to (1.0)

NNP → Berlin (1.0)
PP → TO NNP (1.0)
VP → VP PP (0.3)

| | | | | | |
|---|---|---|---|---|---|
| PRP, NP 1.0*0.4 [0,1] | [0,2] | [0,3] | S 1.0*0.4* 0.7*0.8*0.4*0.1*0.4* 0.9 [0,4] | [0,5] | S 1.0*0.4* 0.7*0.8*0.4*0.1*0.4* 1.0*1.0*1.0* 0.3*0.9 [0,6] |
| | VBP 0.7 [1,2] | [1,3] | VP 0.7* 0.8*0.4*0.1* 0.4 [1,4] | [1,5] | VP 0.7*0.8*0.4*0.1*0.4* 1.0*1.0*1.0* 0.3 [1,6] |
| | | DT 0.8 [2,3] | NP 0.8*0.4* 0.1 [2,4] | [2,5] | [2,6] |
| | | | NN 0.4 [3,4] | [3,5] | [3,6] |
| | | | | TO 1.0 [4,5] | PP 1.0*1.0* 1.0 [4,6] |
| | | | | | NNP 1.0 [5,6] |

I buy a flight to Berlin

0 1 2 3 4 5 6

# Further Reading

- Speech and Language Processing
  - Chapters 12, 13, 14, 15