Syntax Decoding

Hieu Hoang

May 2015

Synchronous Context Free Grammar

- Non-terminal rules

  NP → DET_1 NN_2 JJ_3 # DET_1 JJ_3 NN_2

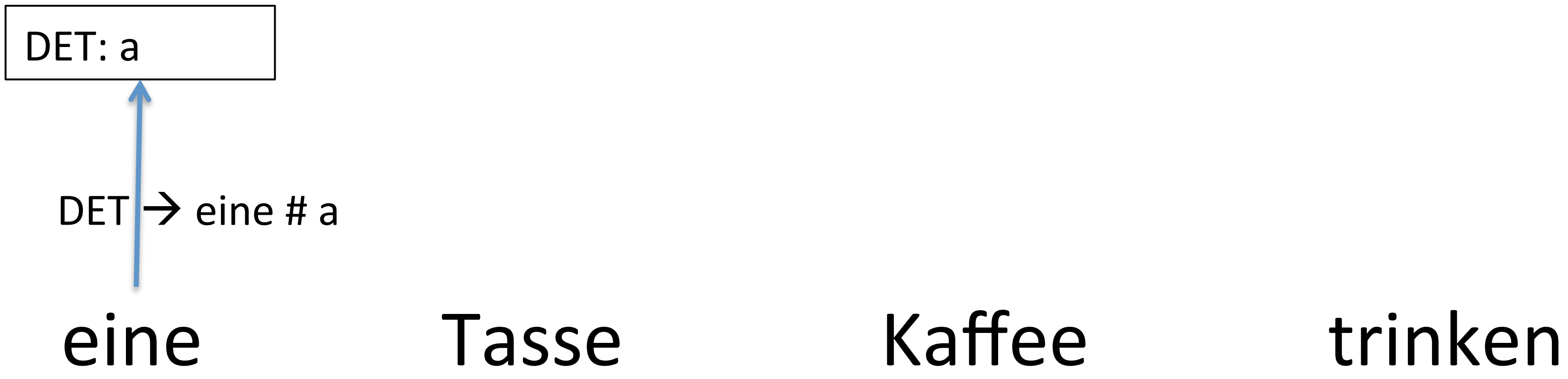
- Terminal rules

  N → maison # house
  NP → la maison blanche # the white house

- Mixed Rules

  NP → la maison JJ_1 # the JJ_1 house

Parsing Algorithm

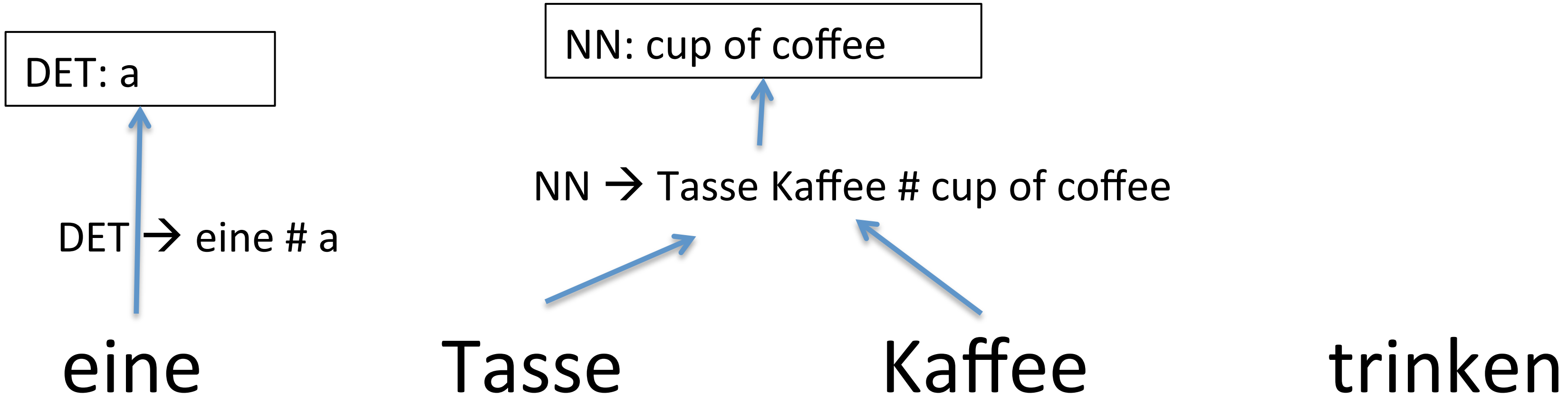


DET: a

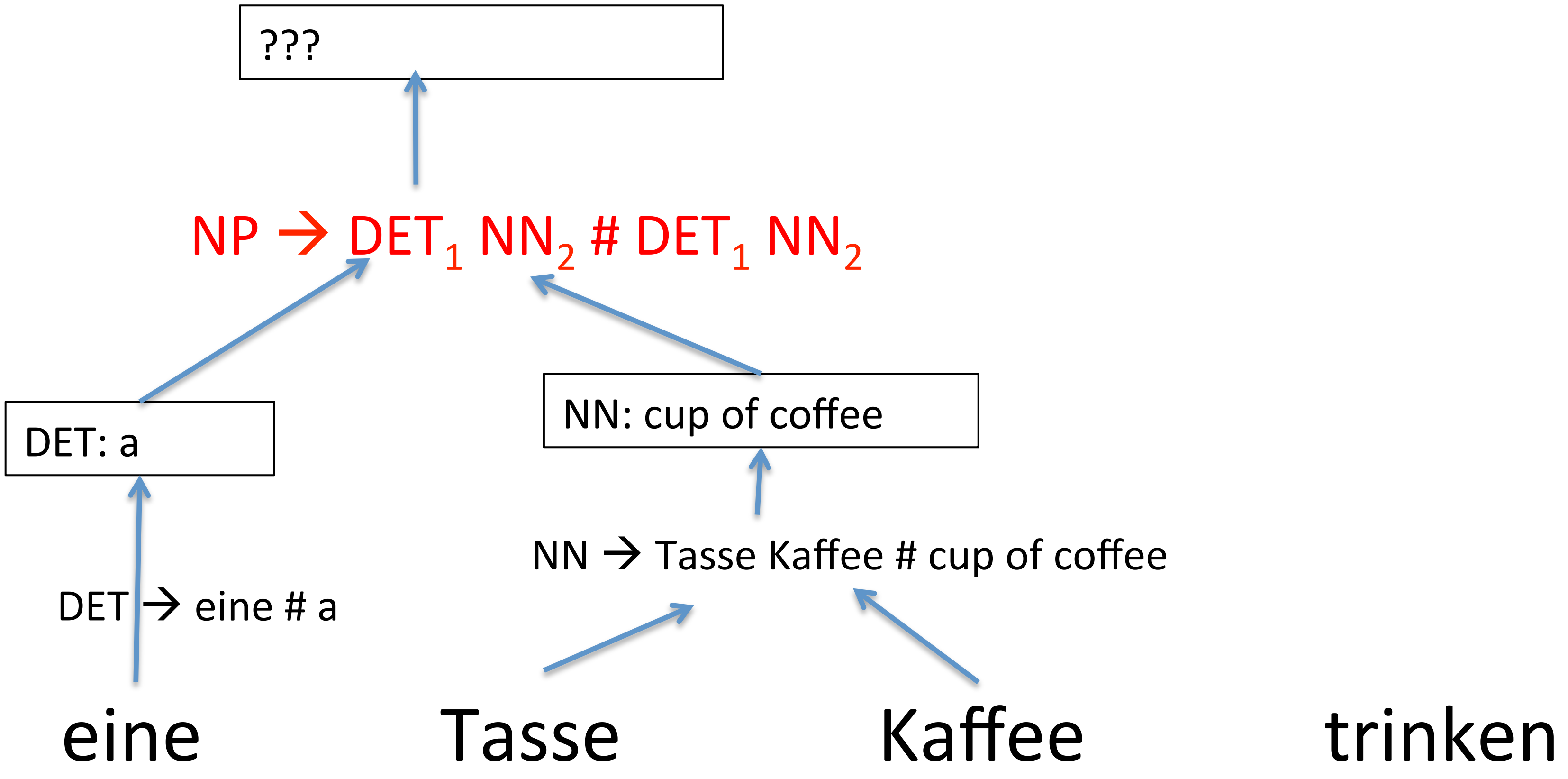
DET → eine # a

eine Tasse Kaffee trinken

Parsing Algorithm



Parsing Algorithm



???

NP → DET_1 NN_2 # DET_1 NN_2

DET: a

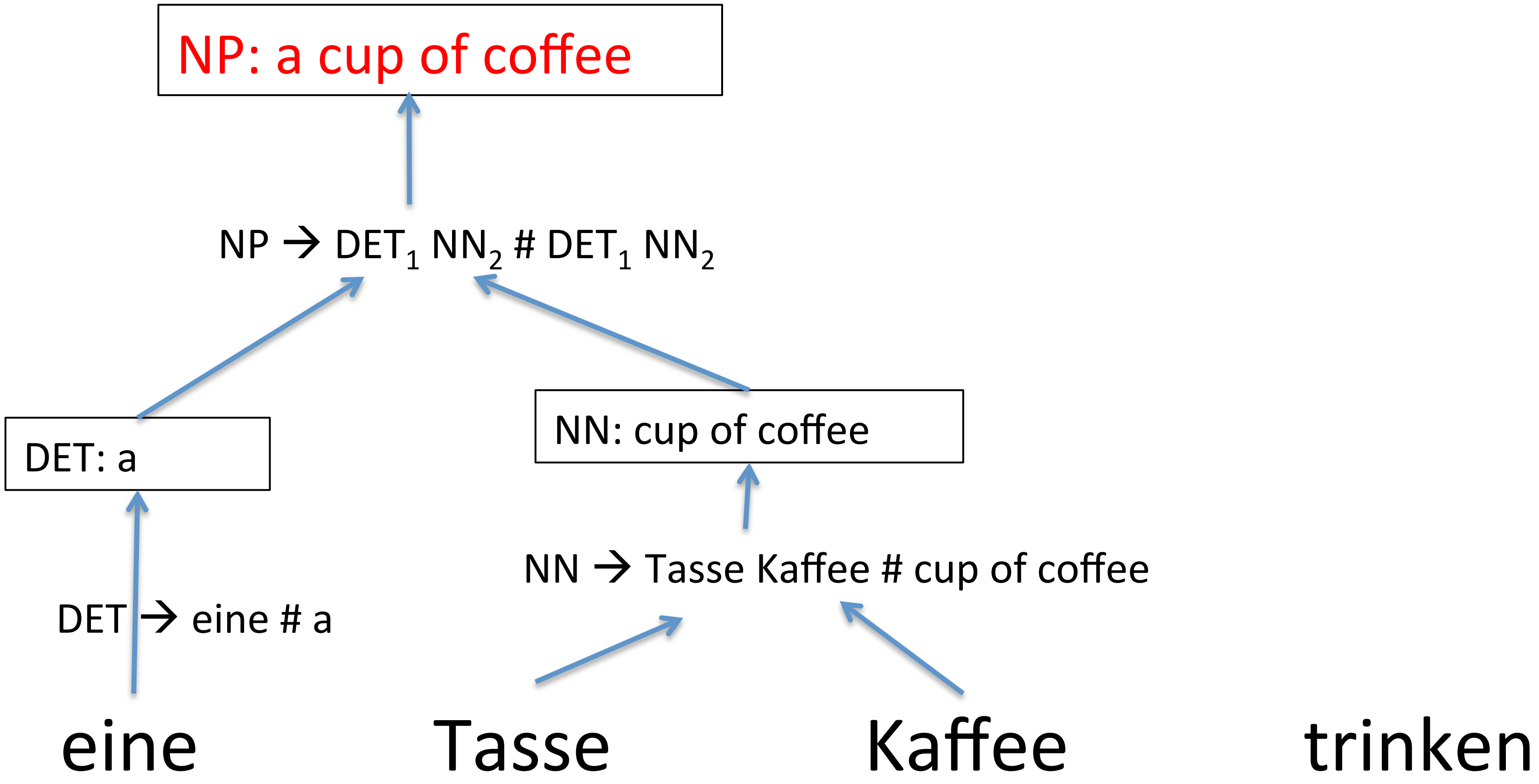
NN: cup of coffee

NN → Tasse Kaffee # cup of coffee

DET → eine # a

eine Tasse Kaffee trinken

Parsing Algorithm



NP: a cup of coffee

NP → DET_1 NN_2 # DET_1 NN_2

DET: a

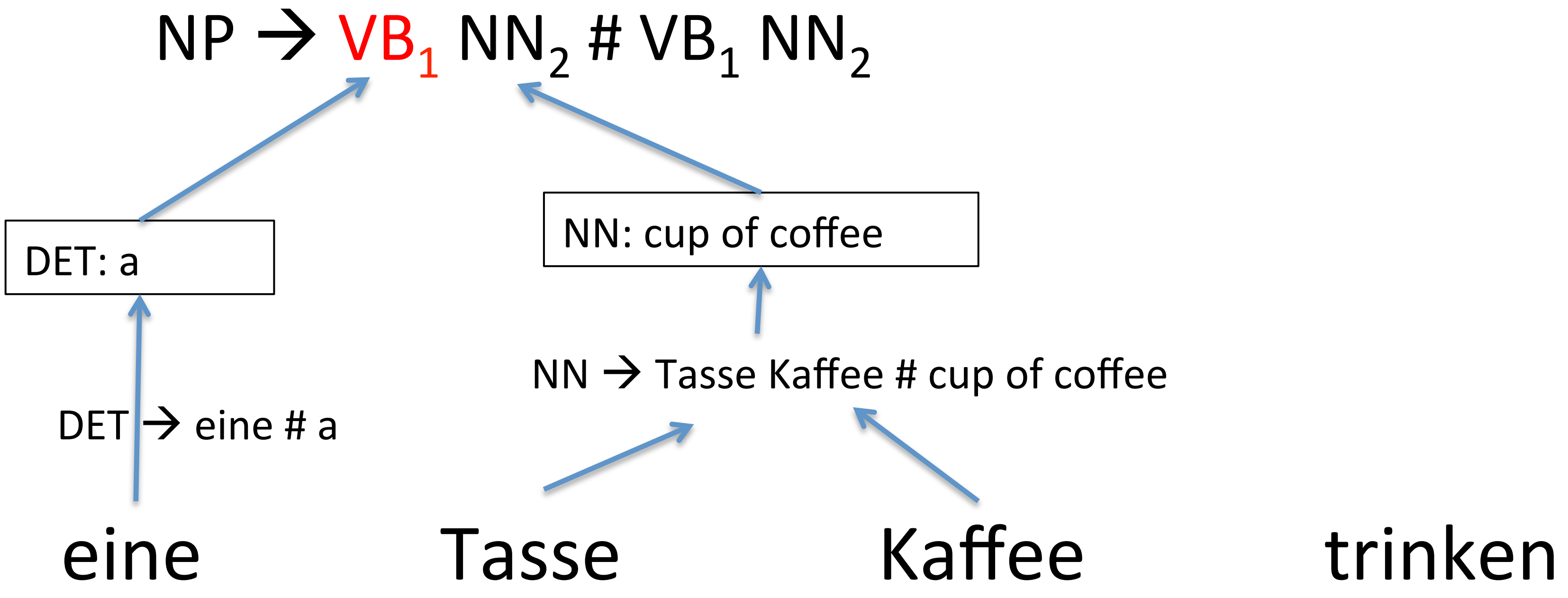
NN: cup of coffee

NN → Tasse Kaffee # cup of coffee

DET → eine # a

eine Tasse Kaffee trinken

Parsing Algorithm



NP → VB_1 NN_2 # VB_1 NN_2

DET: a

NN: cup of coffee

NN → Tasse Kaffee # cup of coffee

DET → eine # a

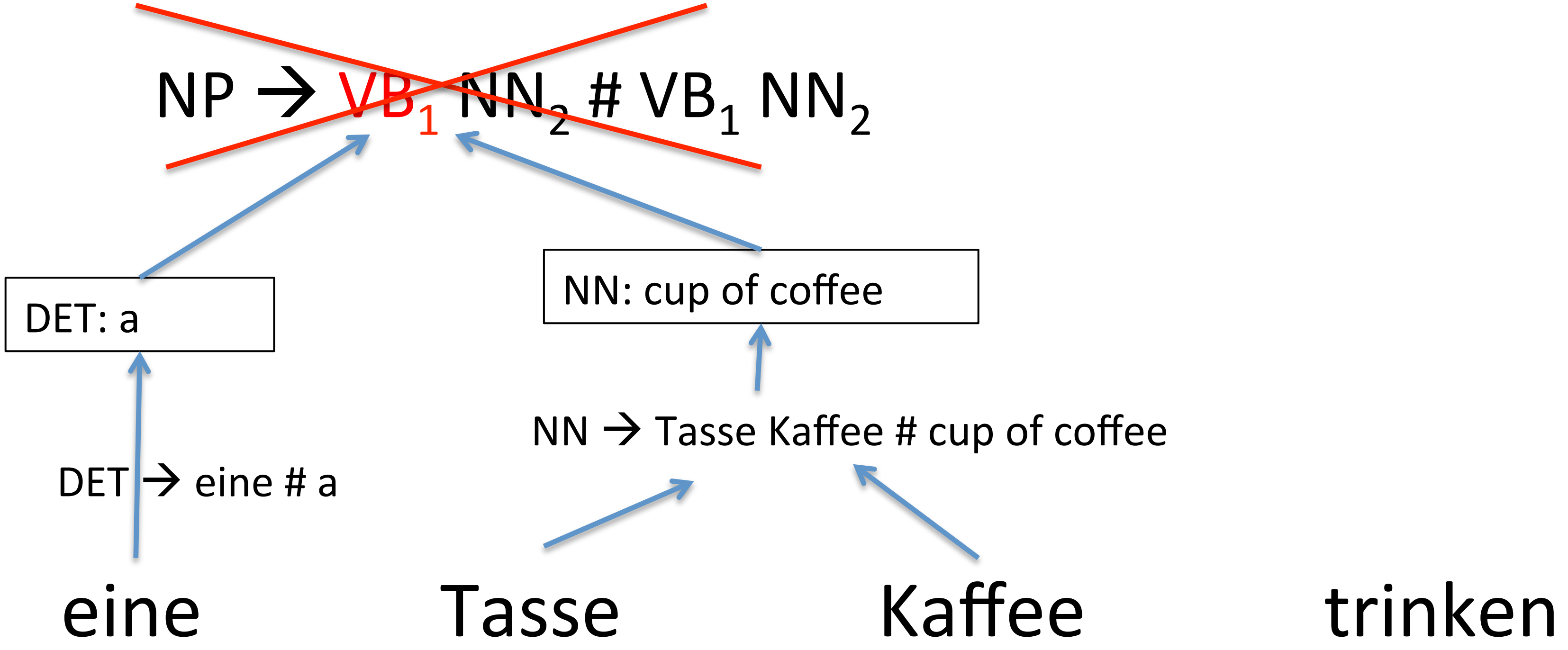
eine

Tasse

Kaffee

trinken

Parsing Algorithm



~~NP → VB_1 NN_2 # VB_1 NN_2~~

DET: a

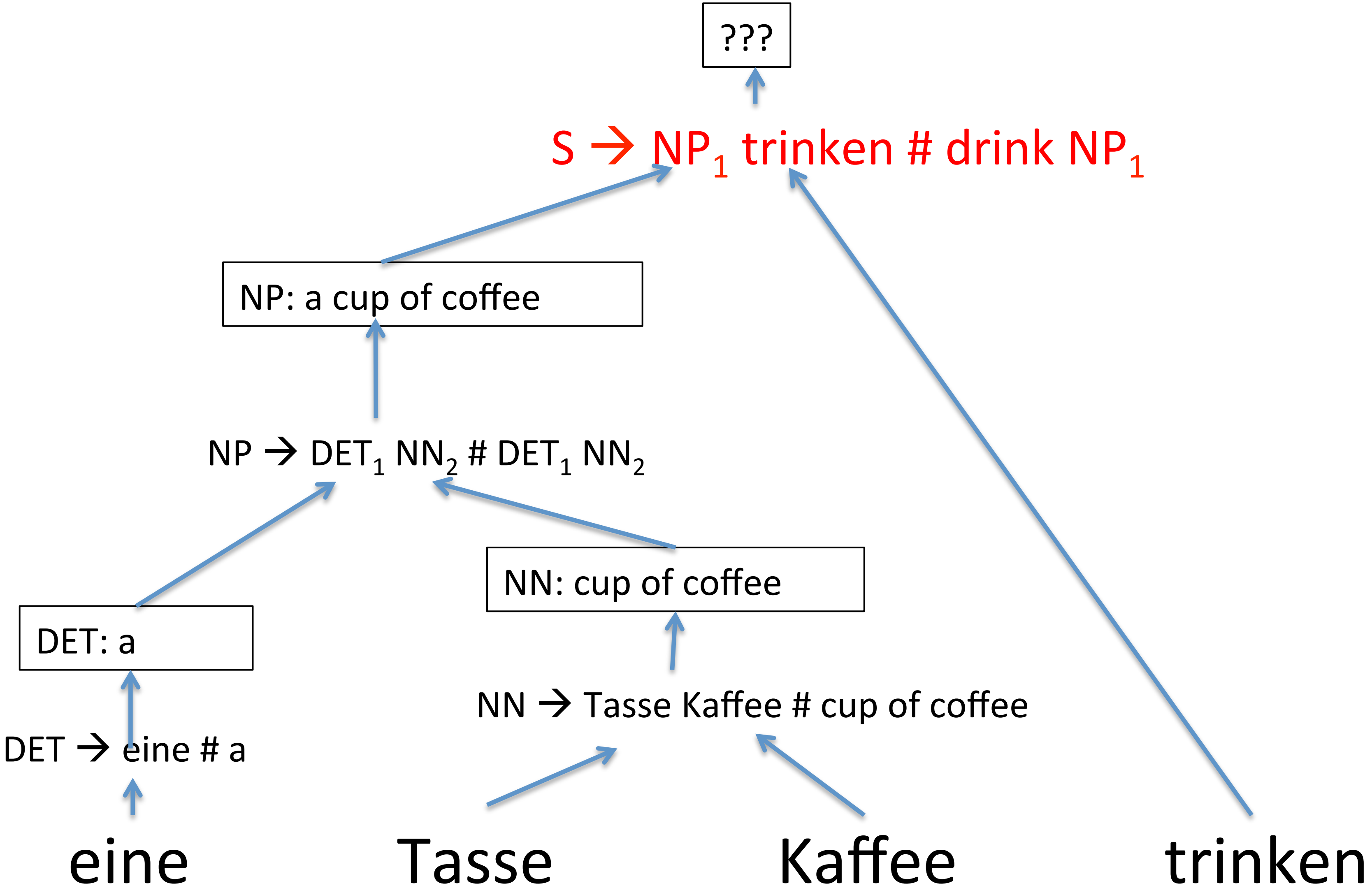
NN: cup of coffee

DET → eine # a

NN → Tasse Kaffee # cup of coffee

eine Tasse Kaffee trinken

Parsing Algorithm



???

S → NP_1 trinken # drink NP_1

NP: a cup of coffee

NP → DET_1 NN_2 # DET_1 NN_2

NN: cup of coffee

DET: a

NN → Tasse Kaffee # cup of coffee

DET → eine # a

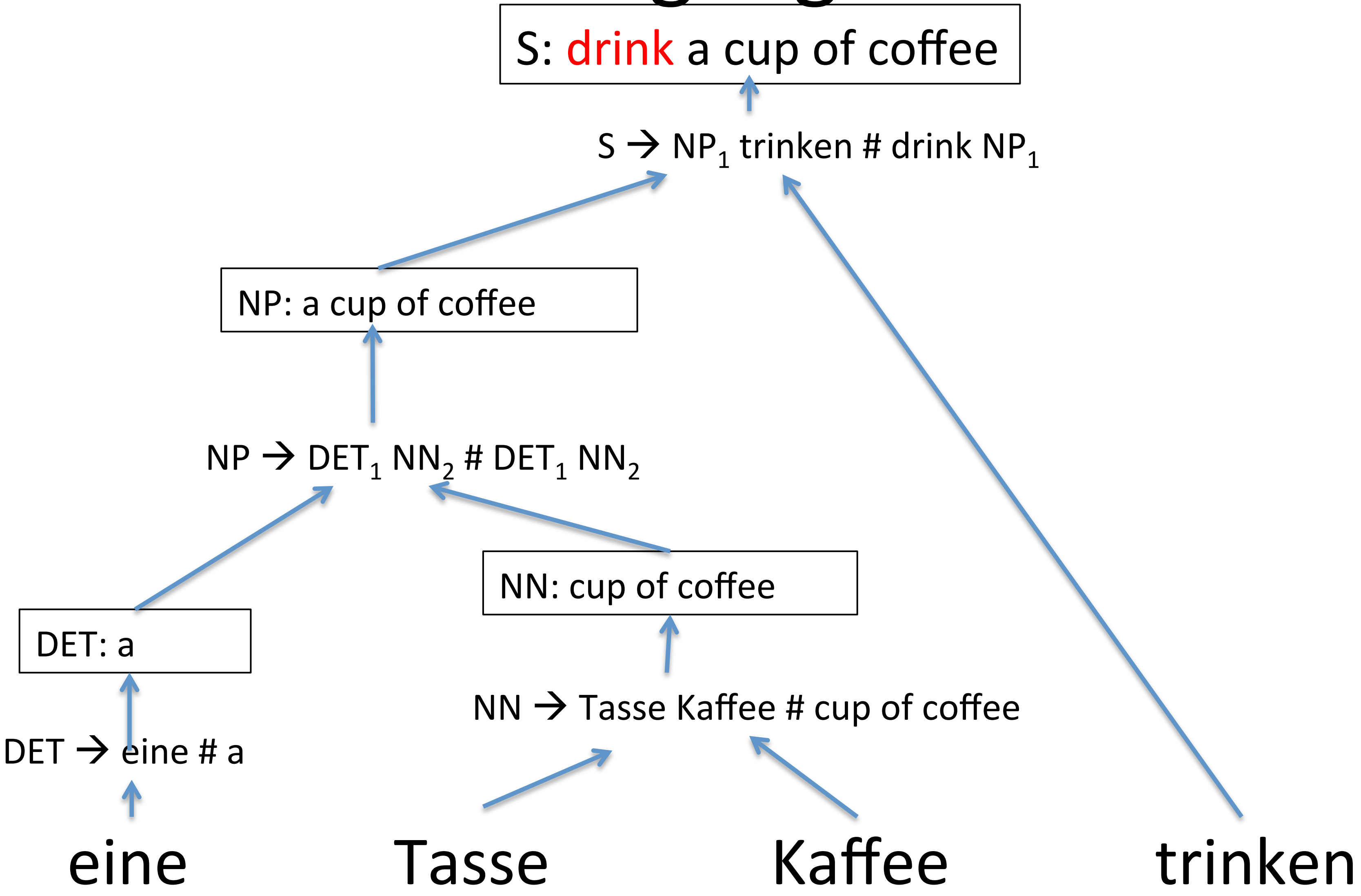
eine

Tasse

Kaffee

trinken

Parsing Algorithm



S: drink a cup of coffee

S → NP_1 trinken # drink NP_1

NP: a cup of coffee

NP → DET_1 NN_2 # DET_1 NN_2

DET: a

NN: cup of coffee

DET → eine # a

NN → Tasse Kaffee # cup of coffee

eine

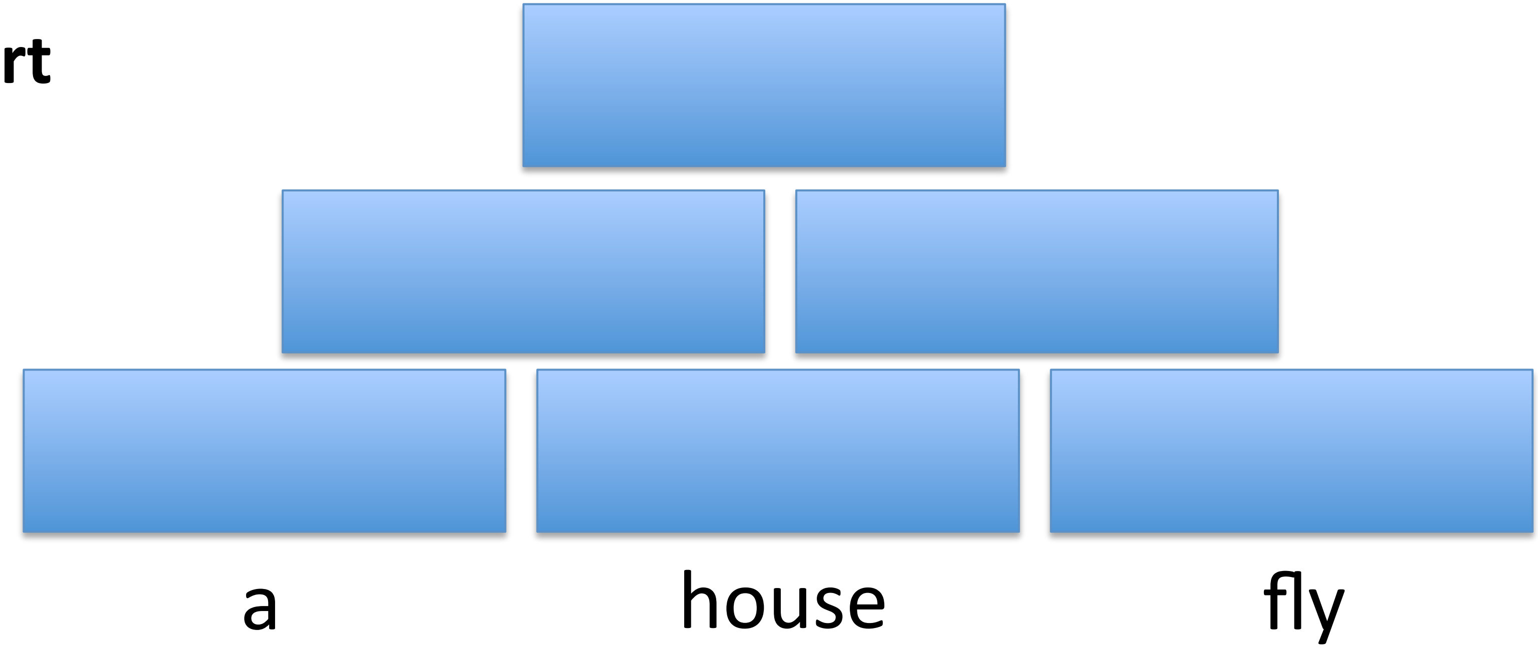
Tasse

Kaffee

trinken

Phrase-table lookup

Chart



- **Lookup ‘a’**

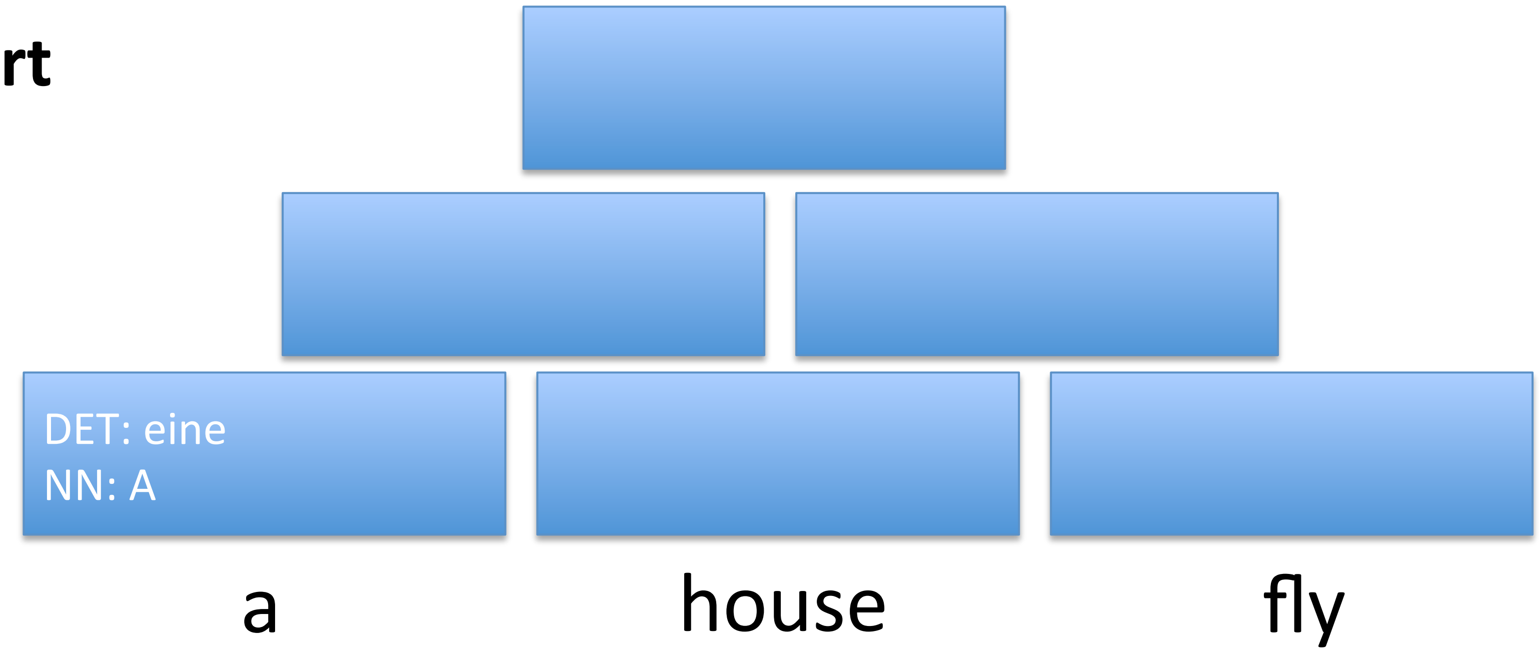
  DET → a # eine

  NN → a # A
- **Create hypotheses**

  DET: eine

  NN: a

Chart



DET: eine

NN: A

a

house

fly

- **Lookup ‘house’**

  NN → house # Haus

  VB → house # beherbergen

  ADJ → house # der Haus

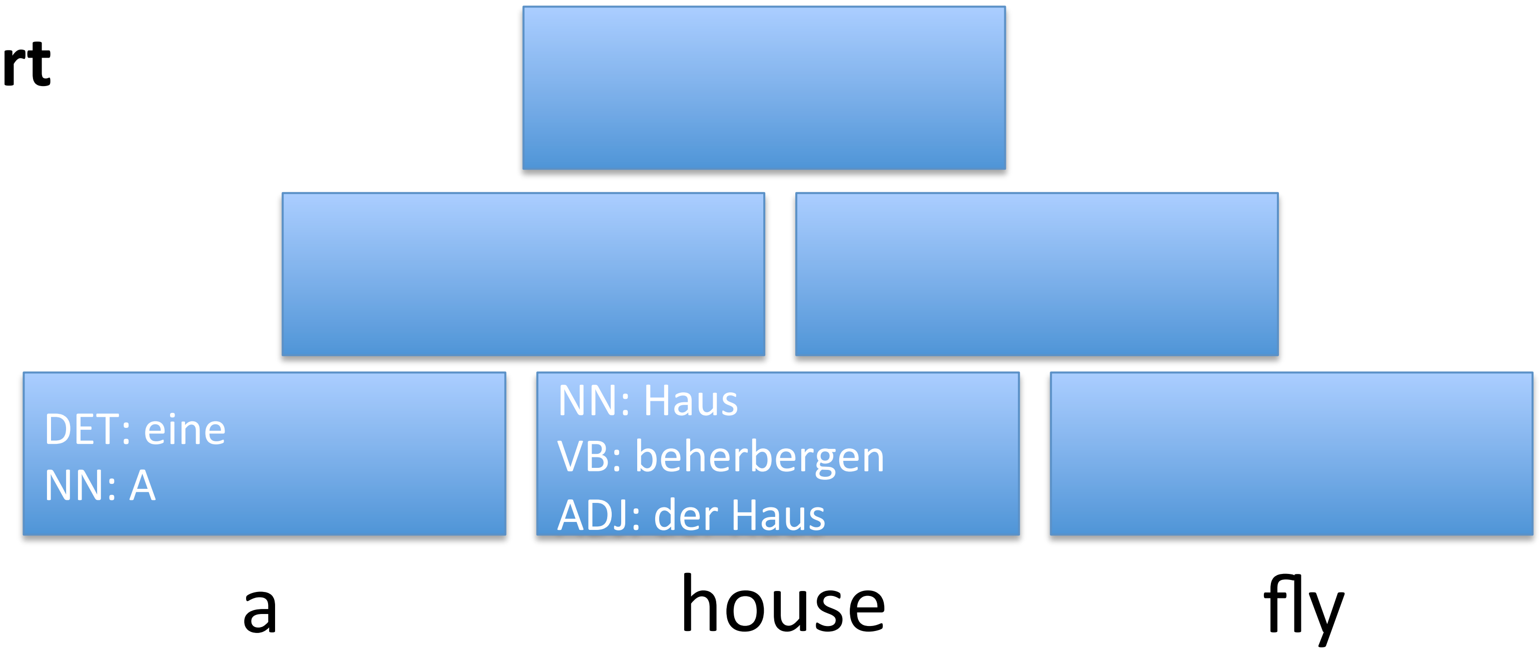
- **Create hypotheses**

  NN: Haus

  VB: beherbergen

  ADJ: der Haus

Chart



DET: eine
NN: A

NN: Haus
VB: beherbergen
ADJ: der Haus

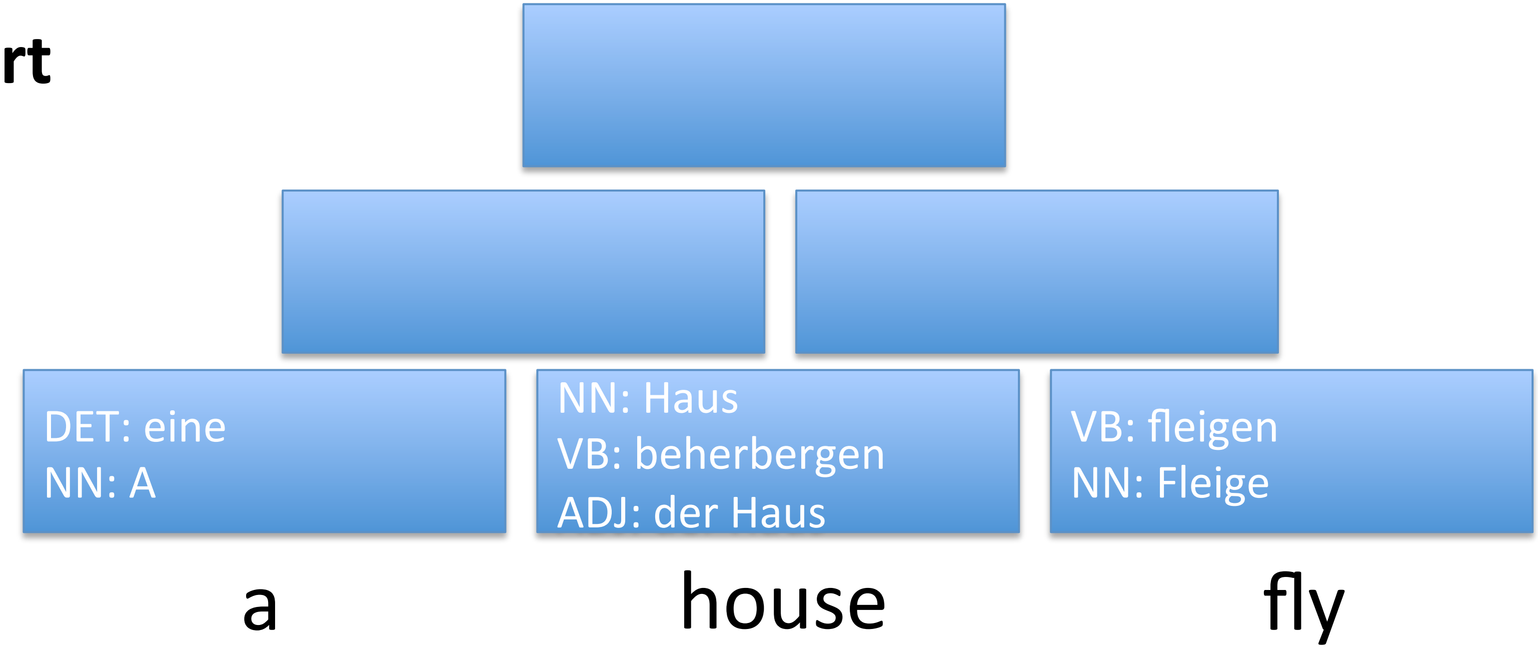
a

house

fly

- **Lookup 'fly'**
  VB → fly # fleigen
  NN → fly # Fliege
- **Create hypotheses**
  VB: fleigen
  NN: Fleige

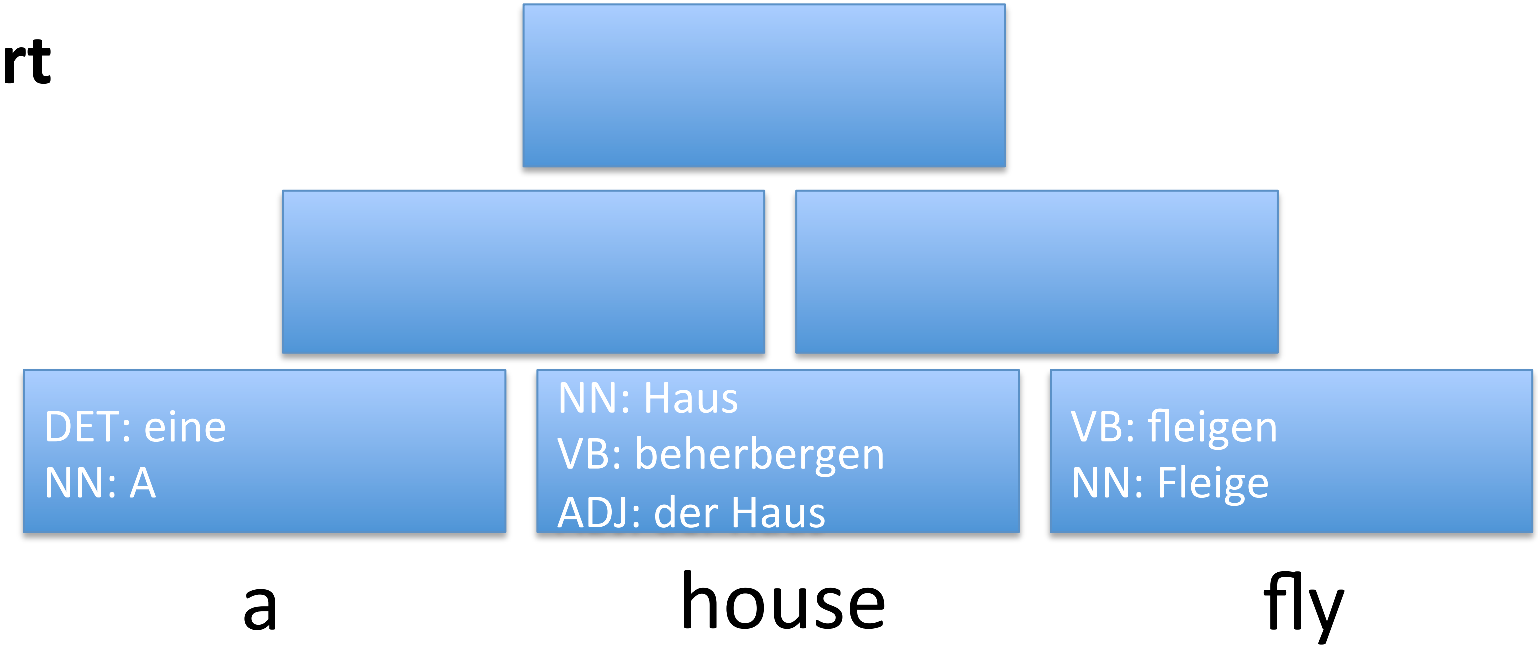
Chart



- **Lookup ‘a house’**

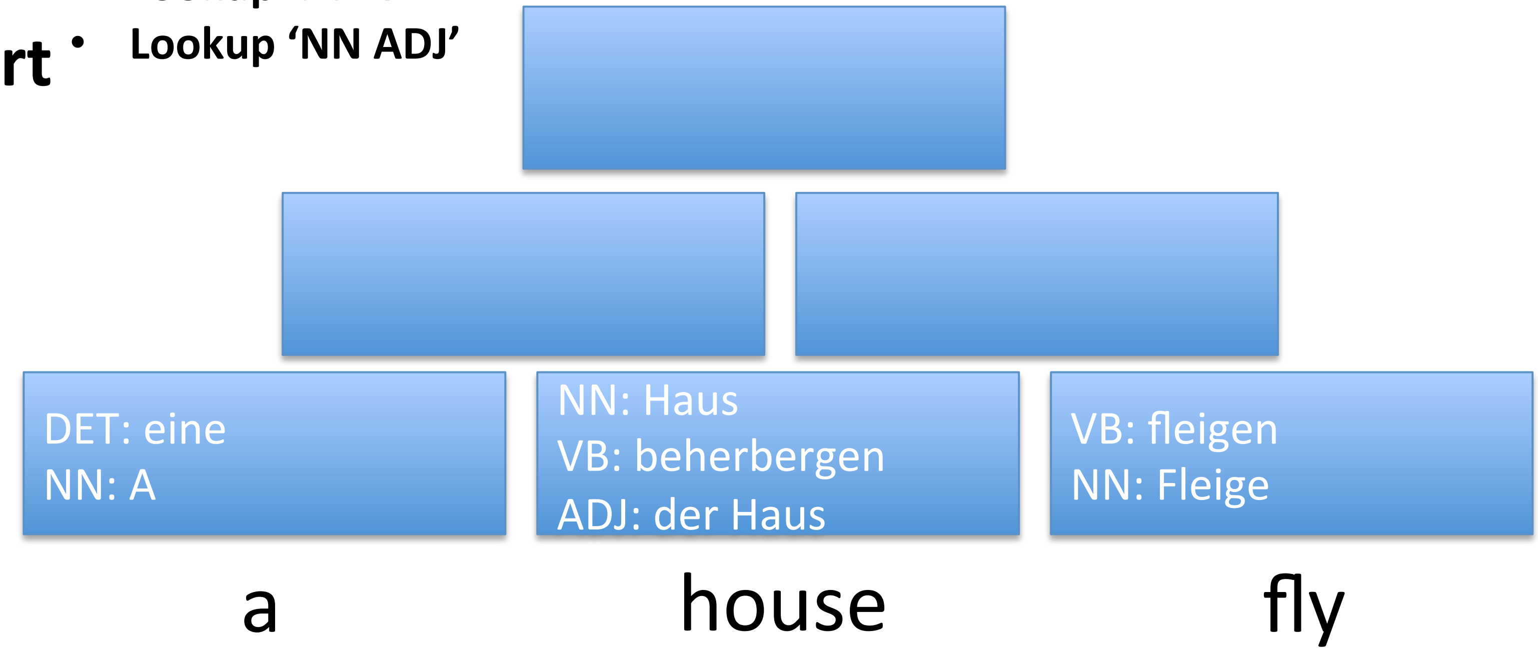
  NP → a house # eine Haus

Chart



- Lookup ‘a house’
- Lookup ‘a NN’
- Lookup ‘a NN’
- Lookup ‘a VB’
- Lookup ‘a ADJ’
- Lookup ‘DET house’
- Lookup ‘DET NN’
- Lookup ‘DET VB’
- Lookup ‘DET ADJ’
- Lookup ‘NN house’
- Lookup ‘NN NN’
- Lookup ‘NN VB’
- Lookup ‘NN ADJ’

Chart



DET: eine
NN: A

NN: Haus
VB: beherbergen
ADJ: der Haus

VB: fleigen
NN: Fleige

a

house

fly

Chart

- Lookup ‘a house’
- Lookup ‘a NN’
- Lookup ‘a NN’
- Lookup ‘a VB’
- Lookup ‘a ADJ’
- Lookup ‘DET house’
- Lookup ‘DET NN’
- Lookup ‘DET VB’
- Lookup ‘DET ADJ’
- Lookup ‘NN house’
- Lookup ‘NN NN’
- Lookup ‘NN VB’
- Lookup ‘NN ADJ’

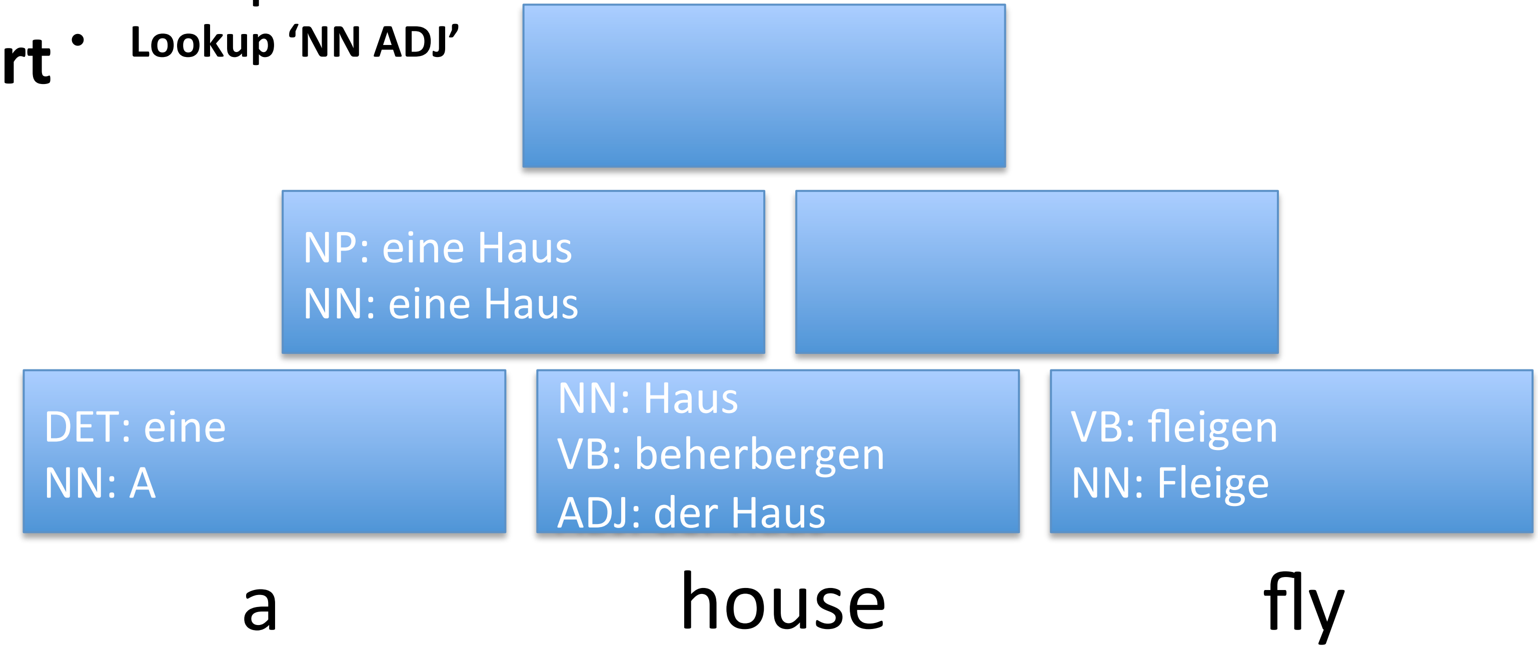
- **Found**

  NP → a house # eine Haus

  NP→ DET_1 NN_2 # DET_1 NN_2

  NP → a NN_1 # eine NN_1

  NN → NN_1 NN_2 # NN_1 NN_2



NP: eine Haus
NN: eine Haus

DET: eine
NN: A

NN: Haus
VB: beherbergen
ADJ: der Haus

VB: fleigen
NN: Fleige

a

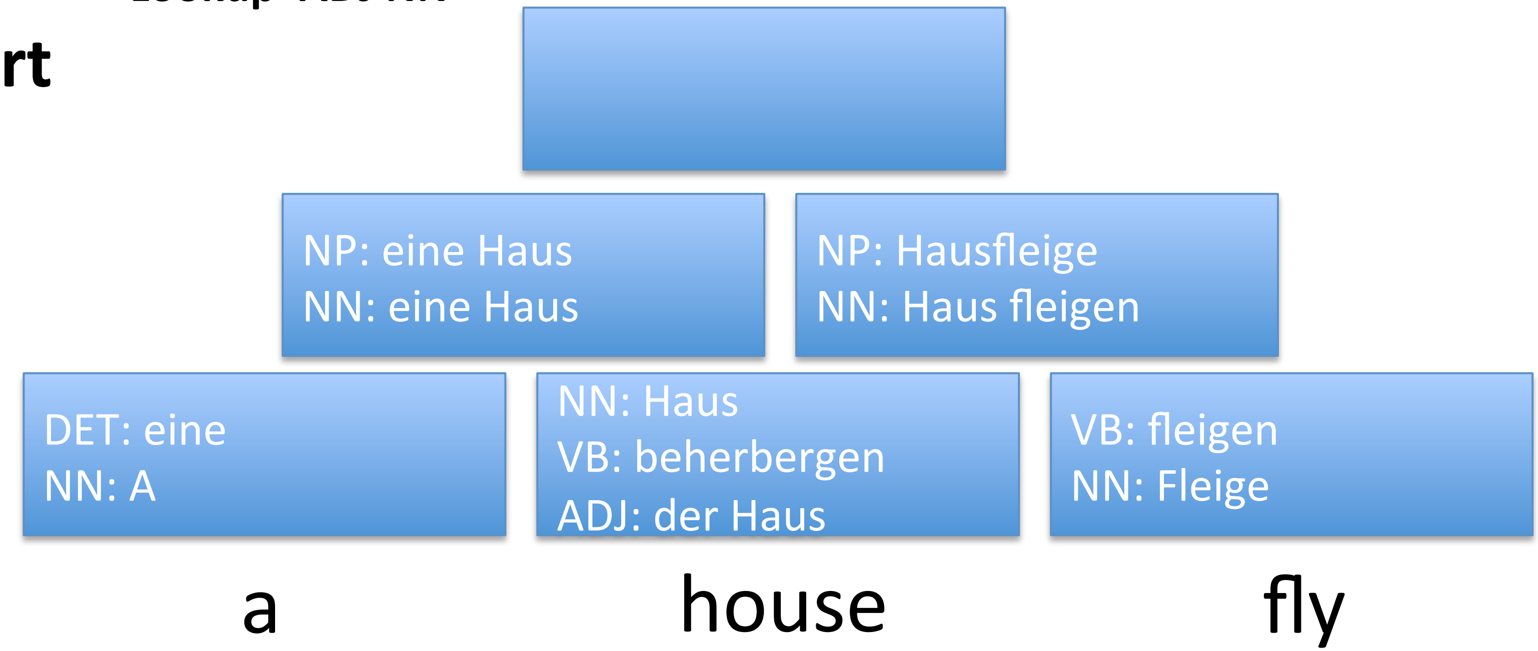
house

fly

- **Lookup ‘house fly’**
- **Lookup ‘house VB’**
- **Lookup ‘house NN’**
- **Lookup ‘NN fly’**
- **Lookup ‘NN VB’**
- **Lookup ‘NN NN’**
- **Lookup ‘VB fly’**
- **Lookup ‘VB VB’**
- **Lookup ‘VB NN’**
- **Lookup ‘ADJ fly’**
- **Lookup ‘ADJ VB’**
- **Lookup ‘ADJ NN’**

- **Found**
  NP → house fly # Hausfliege
  NN → NN_1 NN_2 # NN_1 NN_2

Chart



NP: eine Haus
NN: eine Haus

NP: Hausfleige
NN: Haus fleigen

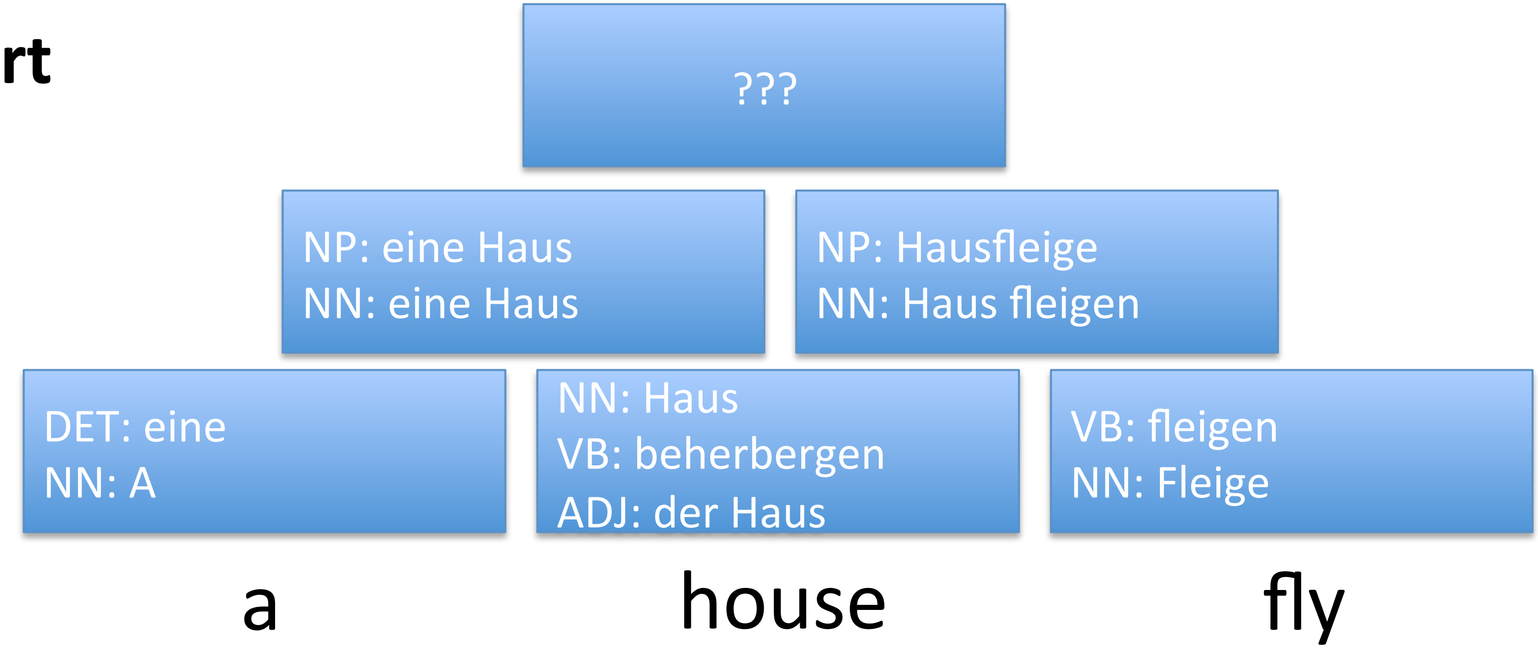
DET: eine
NN: A

NN: Haus
VB: beherbergen
ADJ: der Haus

VB: fleigen
NN: Fleige

a house fly

Chart



???

NP: eine Haus
NN: eine Haus

NP: Hausfleige
NN: Haus fleigen

DET: eine
NN: A

NN: Haus
VB: beherbergen
ADJ: der Haus

VB: fleigen
NN: Fleige

a

house

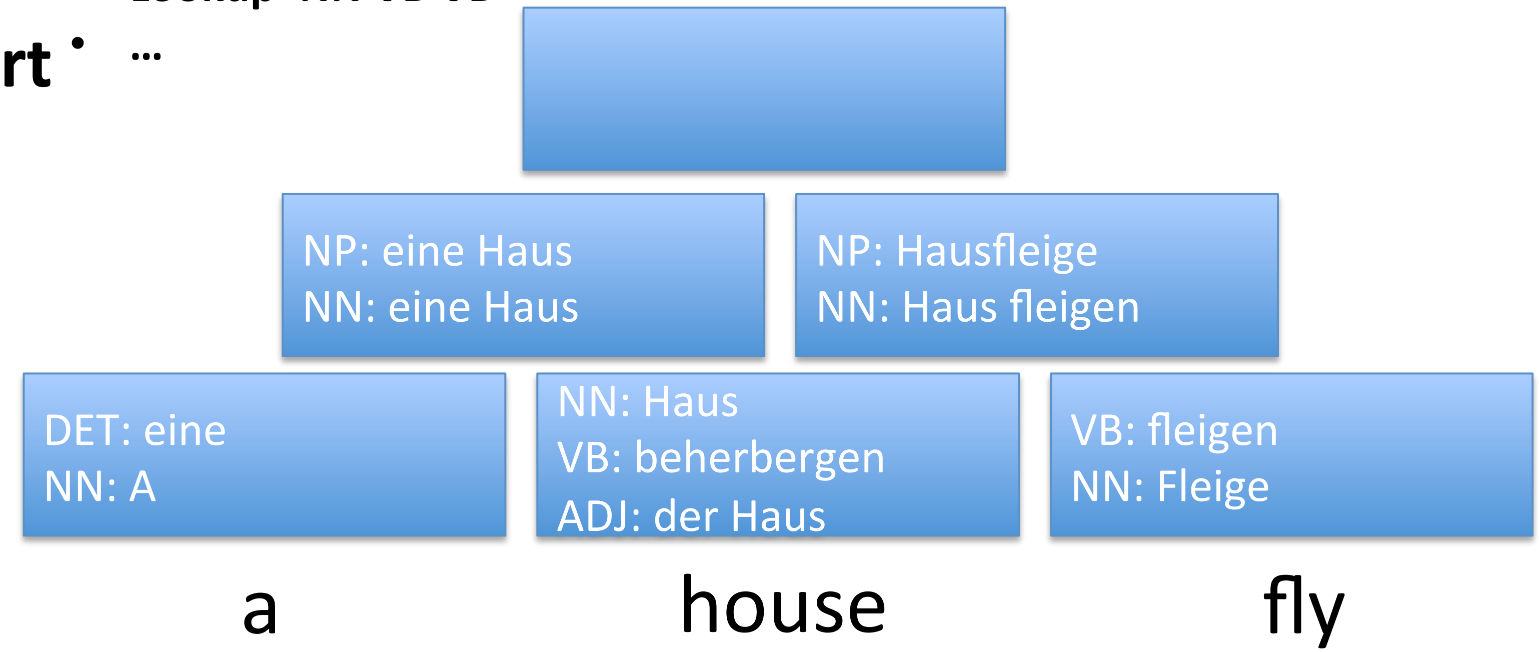
fly

Chart

- Lookup ‘a house fly’
- Lookup ‘a NN fly’
- Lookup ‘a NN VB’
- Lookup ‘a VB NN’
- Lookup ‘a ADJ fly’
- Lookup ‘DET house VB’
- Lookup ‘DET NN NN’
- Lookup ‘DET VB fly’
- Lookup ‘DET ADJ VB’
- Lookup ‘NN house NN’
- Lookup ‘NN NN fly’
- Lookup ‘NN VB VB’
- …

- Lookup ‘a NP’
- Lookup ‘a NN’
- Lookup ‘DET’
- …

- Lookup ‘NP fly’
- Lookup ‘NP VB’
- Lookup ‘NP NN’
- …



NP: eine Haus
NN: eine Haus

NP: Hausfleige
NN: Haus fleigen

DET: eine
NN: A

NN: Haus
VB: beherbergen
ADJ: der Haus

VB: fleigen
NN: Fleige

Cocke–Younger–Kasami (CYK)

- Efficient parsing of CFG
- Only grammar in Chomsky Normal Form (CNF)
  - A→ eats
  - A → B C
- Not for Machine Translation
  - Not CNF grammar
  - Rules with 2+ non-terminals
  - Rules with terminals AND non-terminals
- CYK+
  - By Chappelier and Rajman (1998)

CKY+

- Intuition:
  - If we need

    q → A B C # x

    then prefix <u>must</u> exist

    q → A B # y
- Bottom-up parsing
- Non-Chomsky Normal Form

CYK+

Grammar

DET → a # eine

NN → a # A

NN → house # Haus

VB → house # beherbergen

ADJ → house # der Haus

NN → house fly # Hausfliege

NN → fly # Fliege

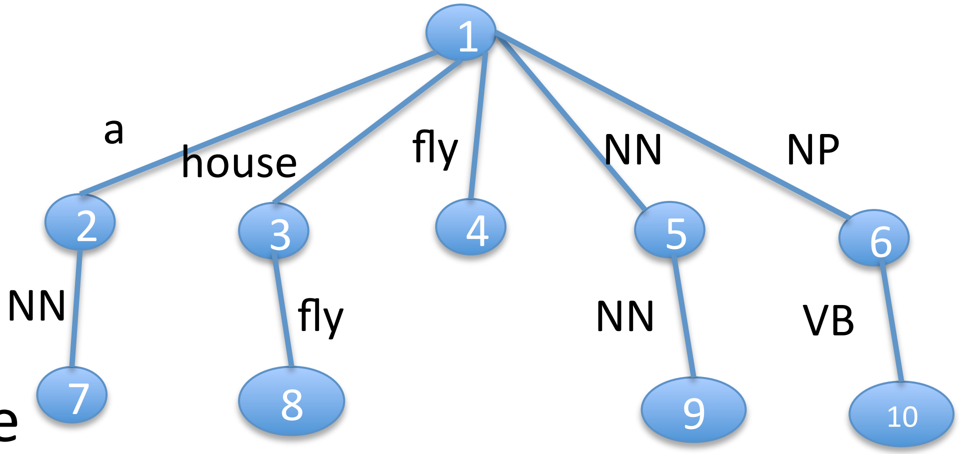
VB → fly # fliegen

NP → a NN_1 # eine NN_1

NN → NN_1 NN_2 # NN_1 NN_2

S → NP_1 VB_2 # NP_1 VB_2

Trie



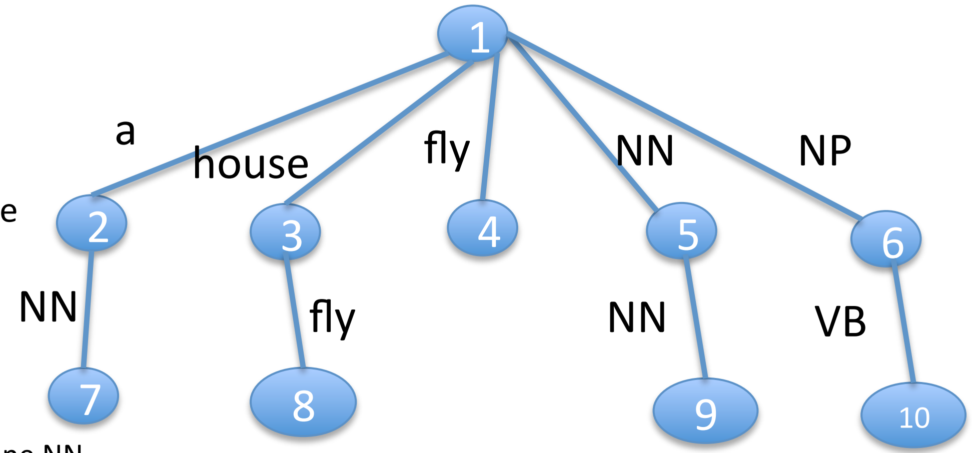
Does ‘a house fly’ exist?

‘a house’ does NOT exist

→ ‘a house fly’ NOT exist

CYK+

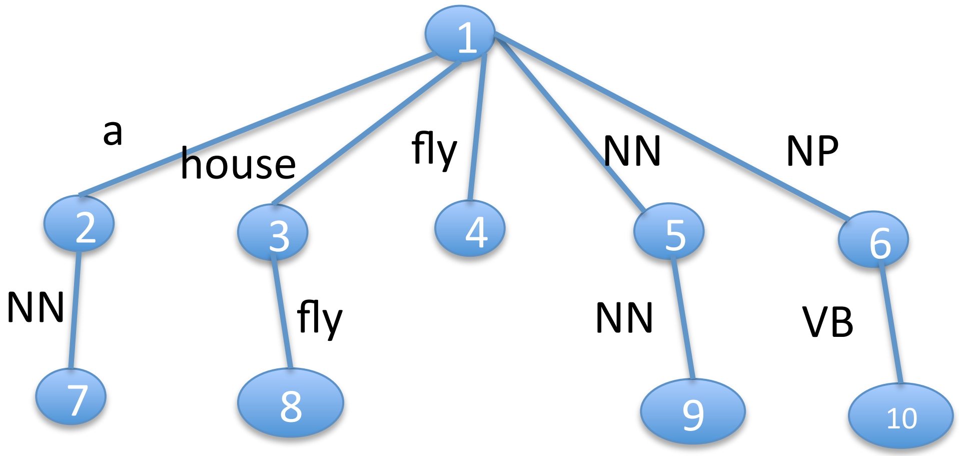
Trie



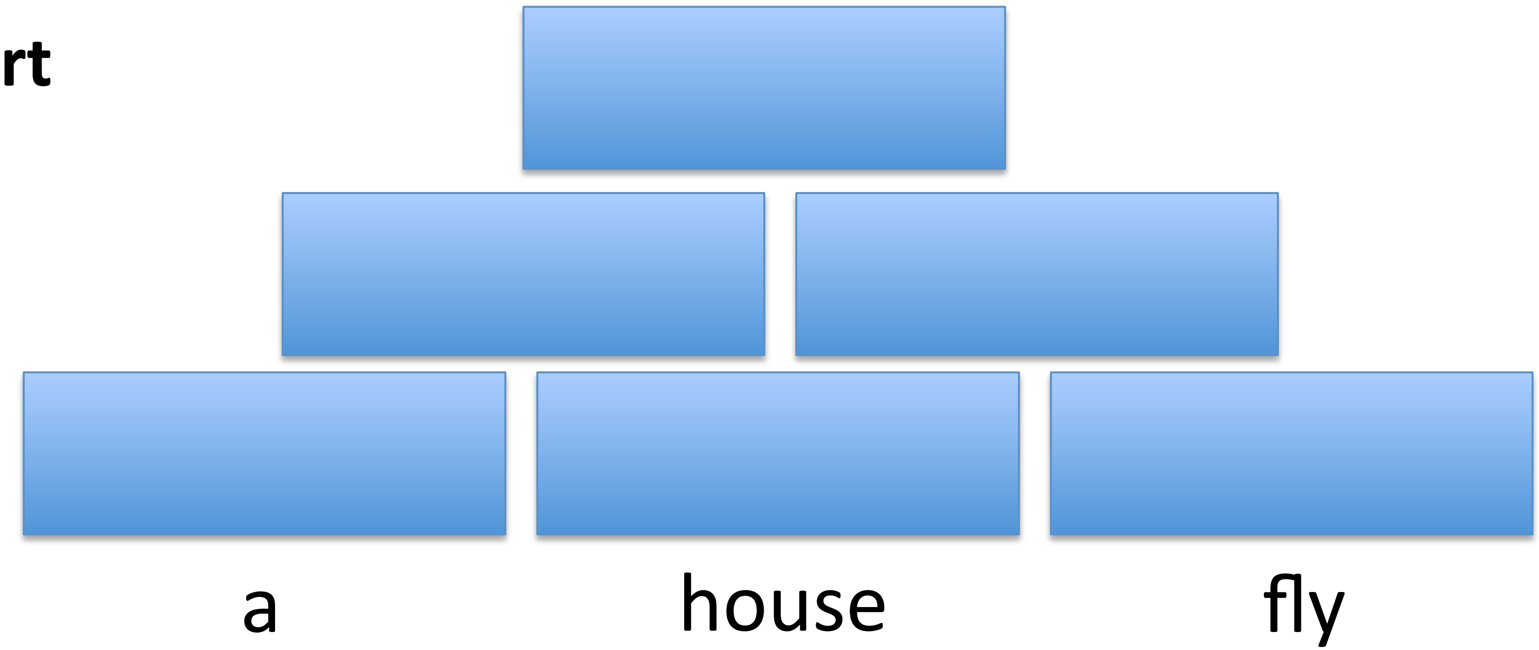
DET → a # eine
NN → a # A

NN → a NN_2 # eine NN_2

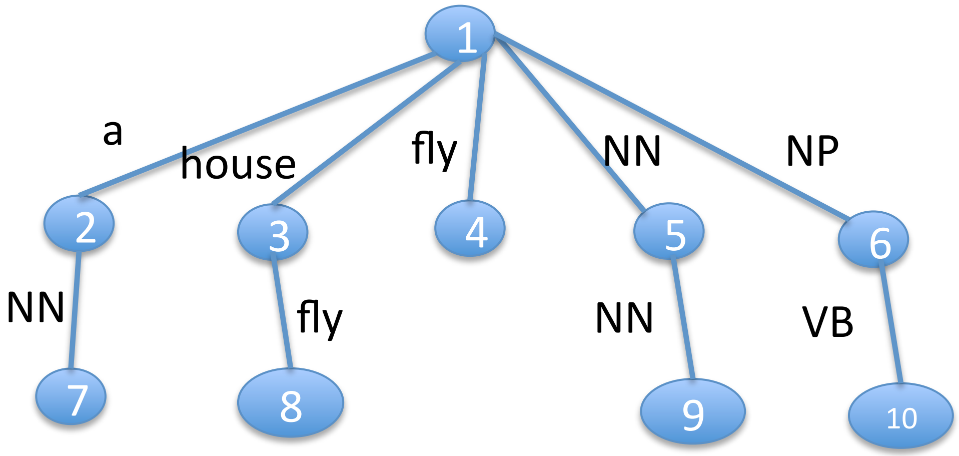
Trie



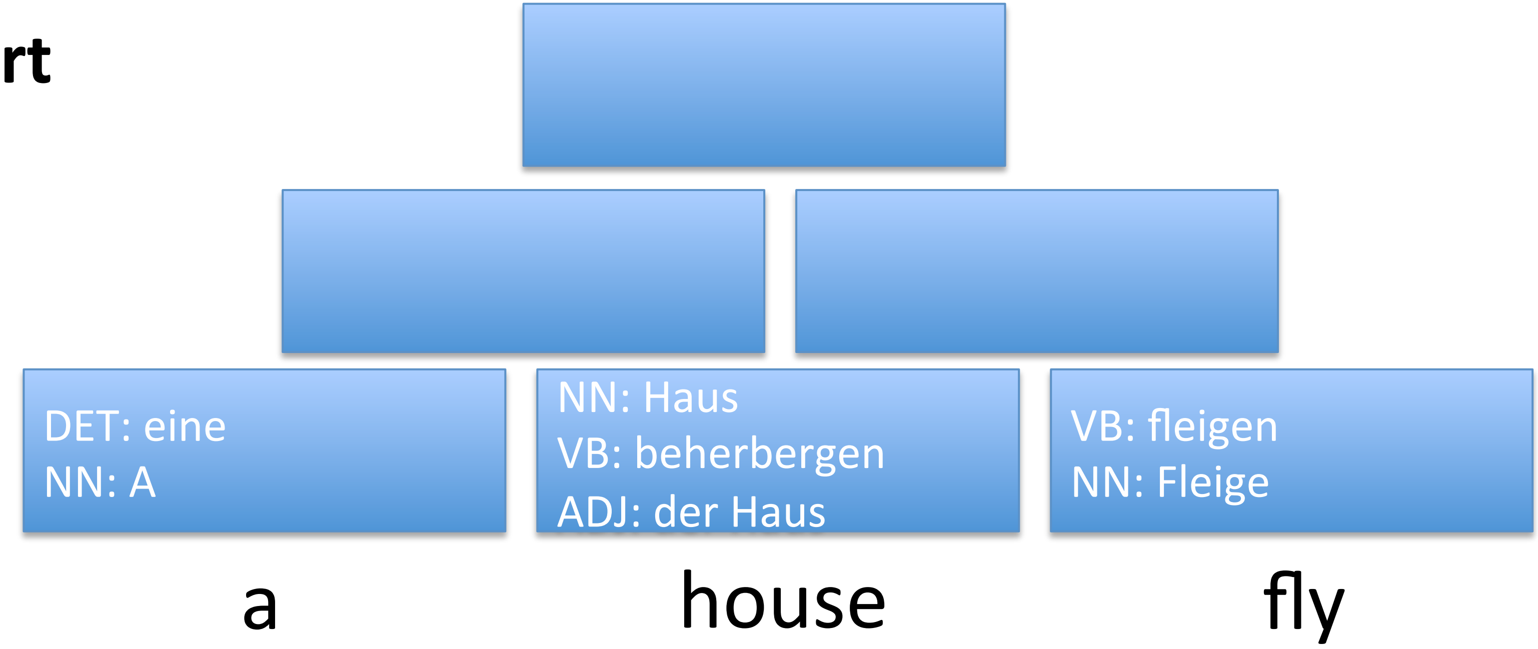
Chart



Trie



Chart


DET: eine
NN: A

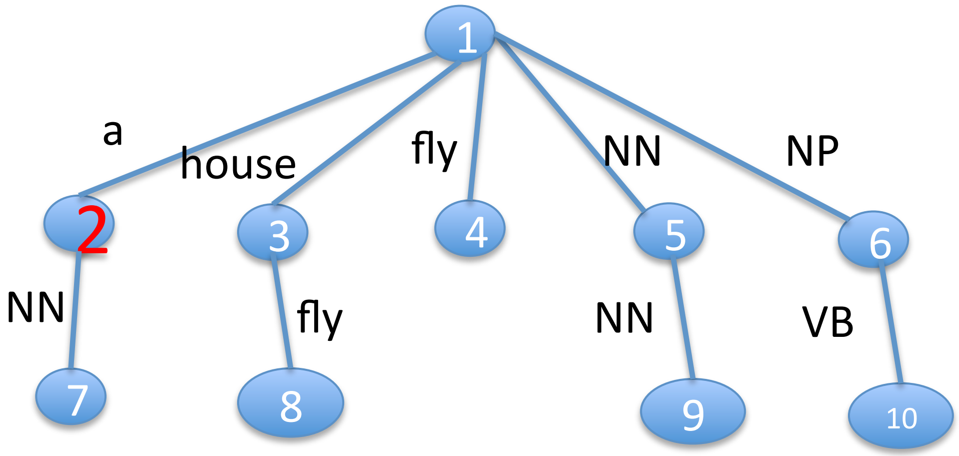
NN: Haus
VB: beherbergen
ADJ: der Haus

VB: fleigen
NN: Fleige

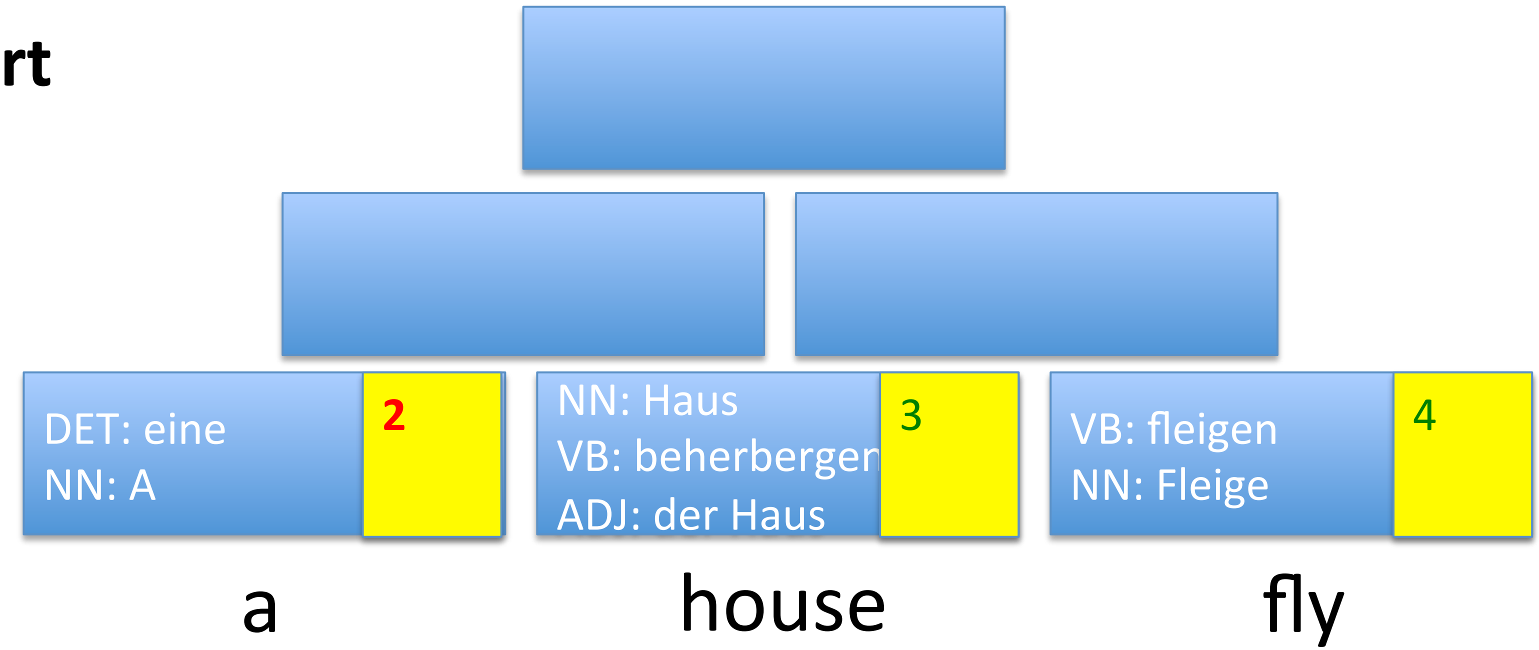
a house fly

DO Look up all single words in trie

Trie



Chart



DET: eine
NN: A
2

NN: Haus
VB: beherbergen
ADJ: der Haus
3

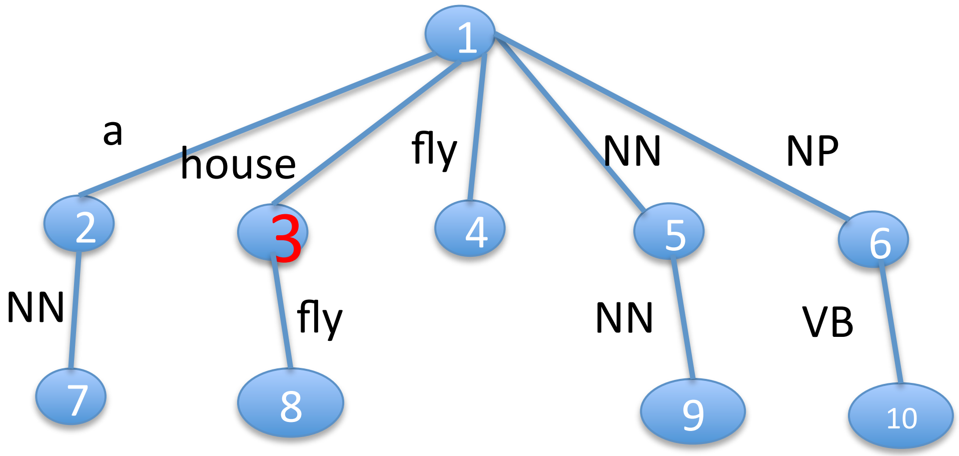
VB: fleigen
NN: Fleige
4

a

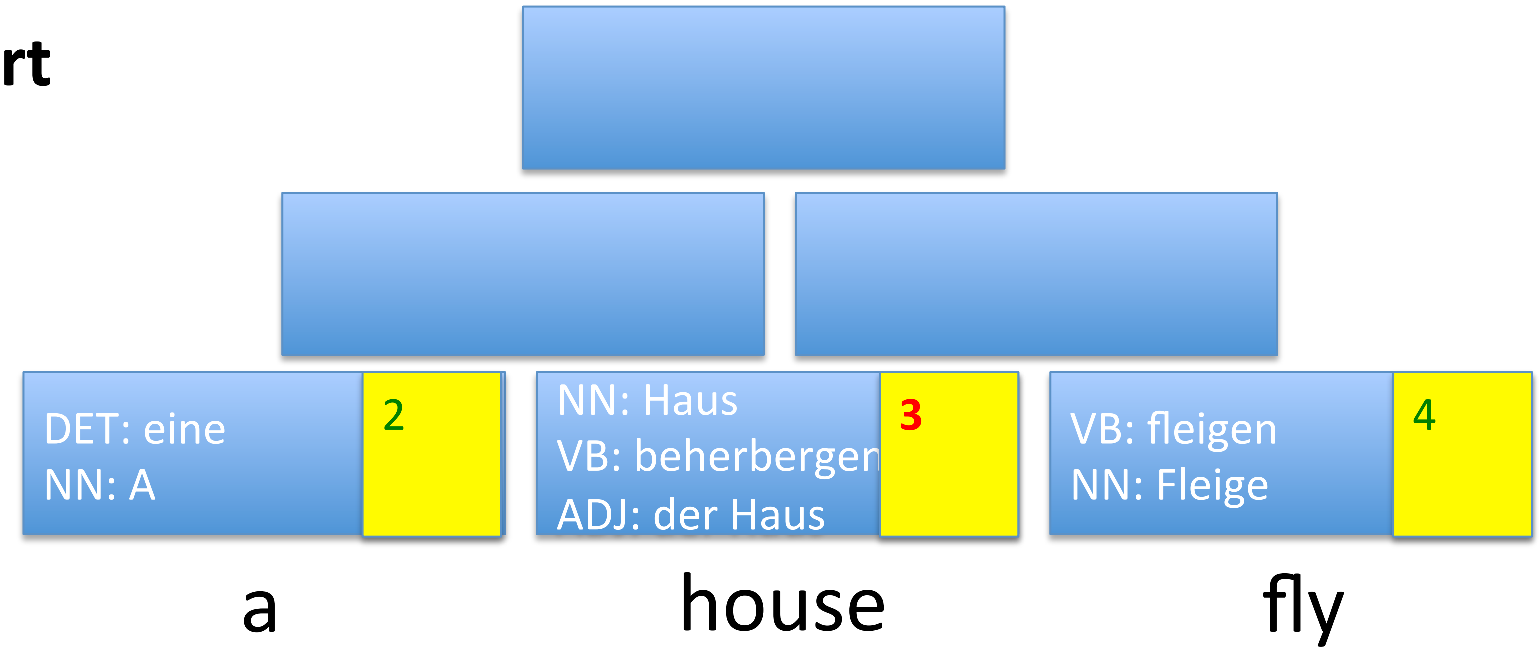
house

fly

Trie



Chart



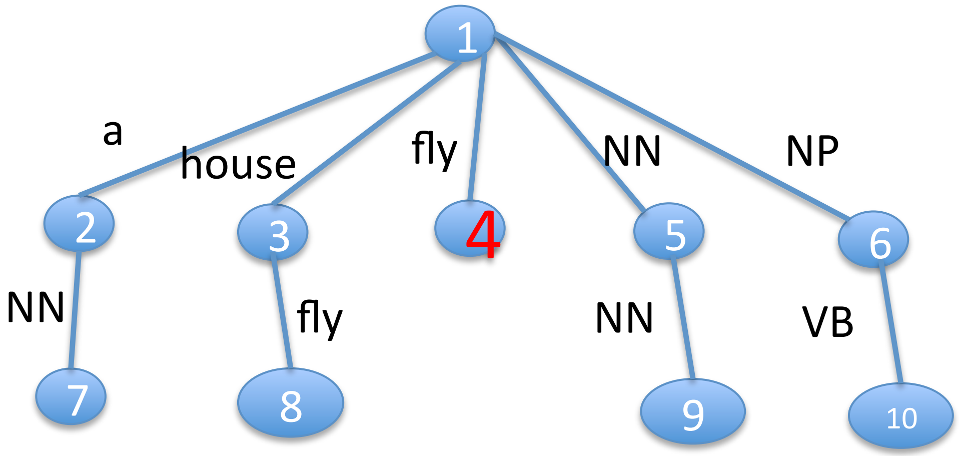
DET: eine
NN: A
2

NN: Haus
VB: beherbergen
ADJ: der Haus
3

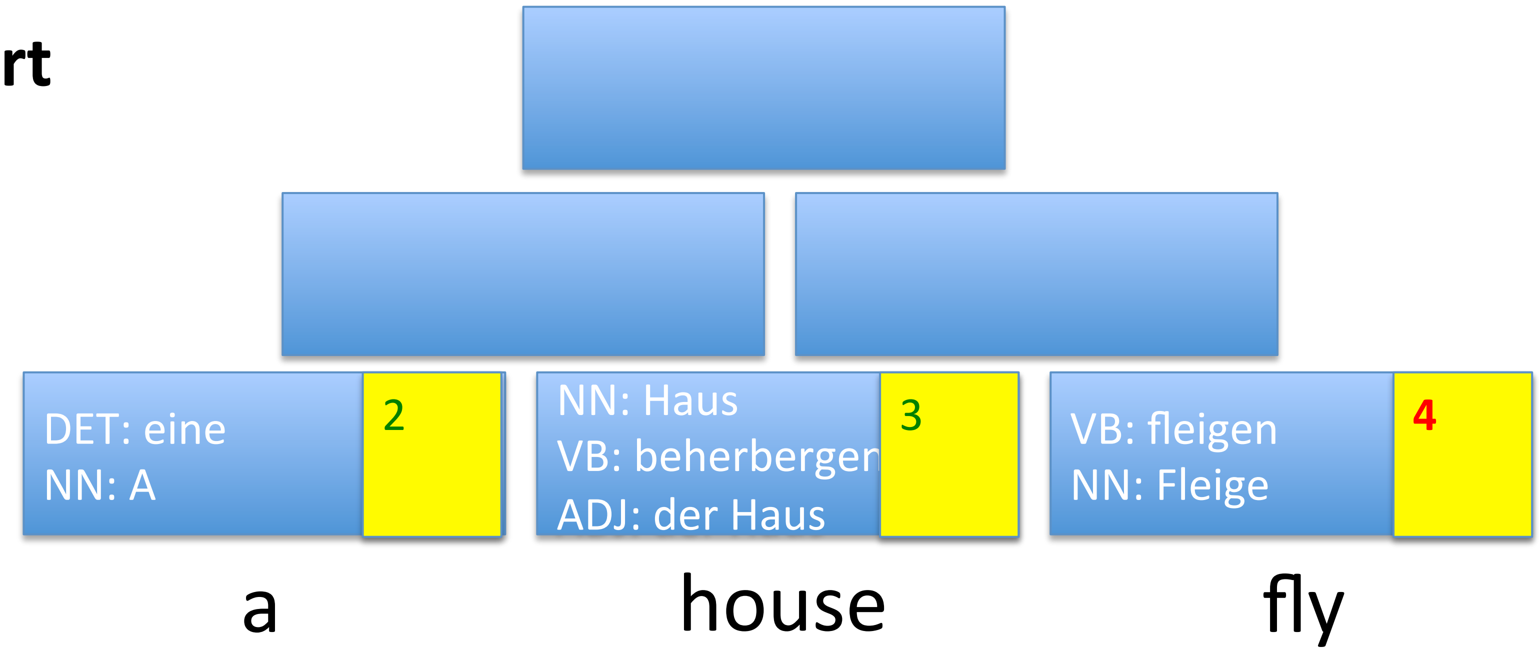
VB: fleigen
NN: Fleige
4

a house fly

Trie



Chart


DET: eine
NN: A

2

NN: Haus
VB: beherbergen
ADJ: der Haus

3

VB: fleigen
NN: Fleige

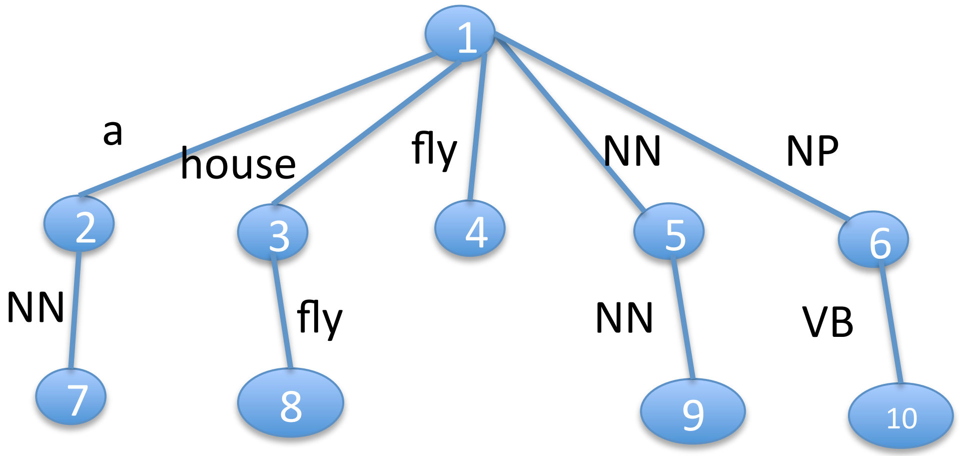
4

a

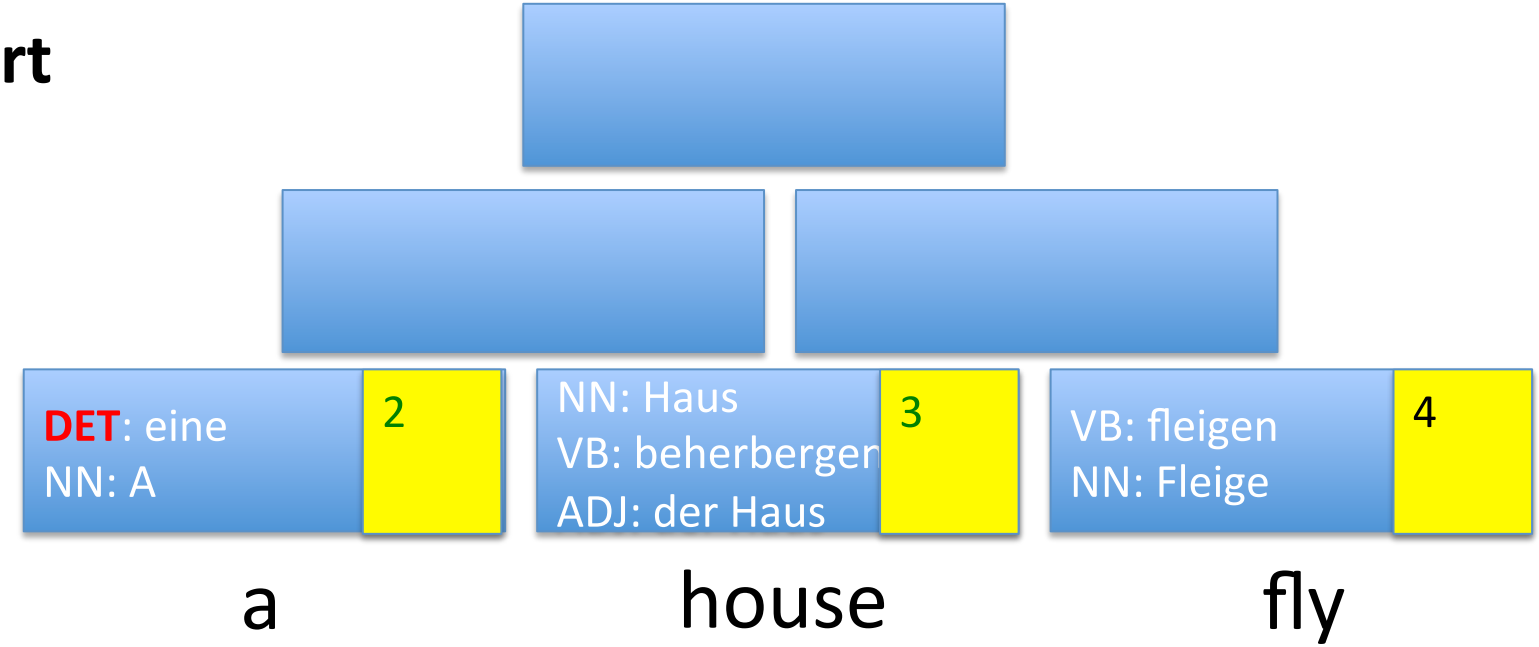
house

fly

Trie



Chart



DET: eine
NN: A

2

NN: Haus
VB: beherbergen
ADJ: der Haus

3

VB: fleigen
NN: Fleige

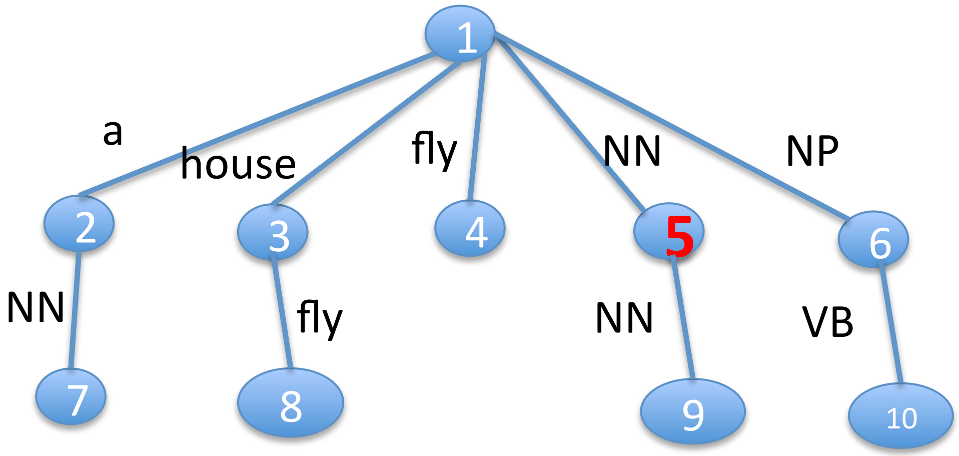
4

a

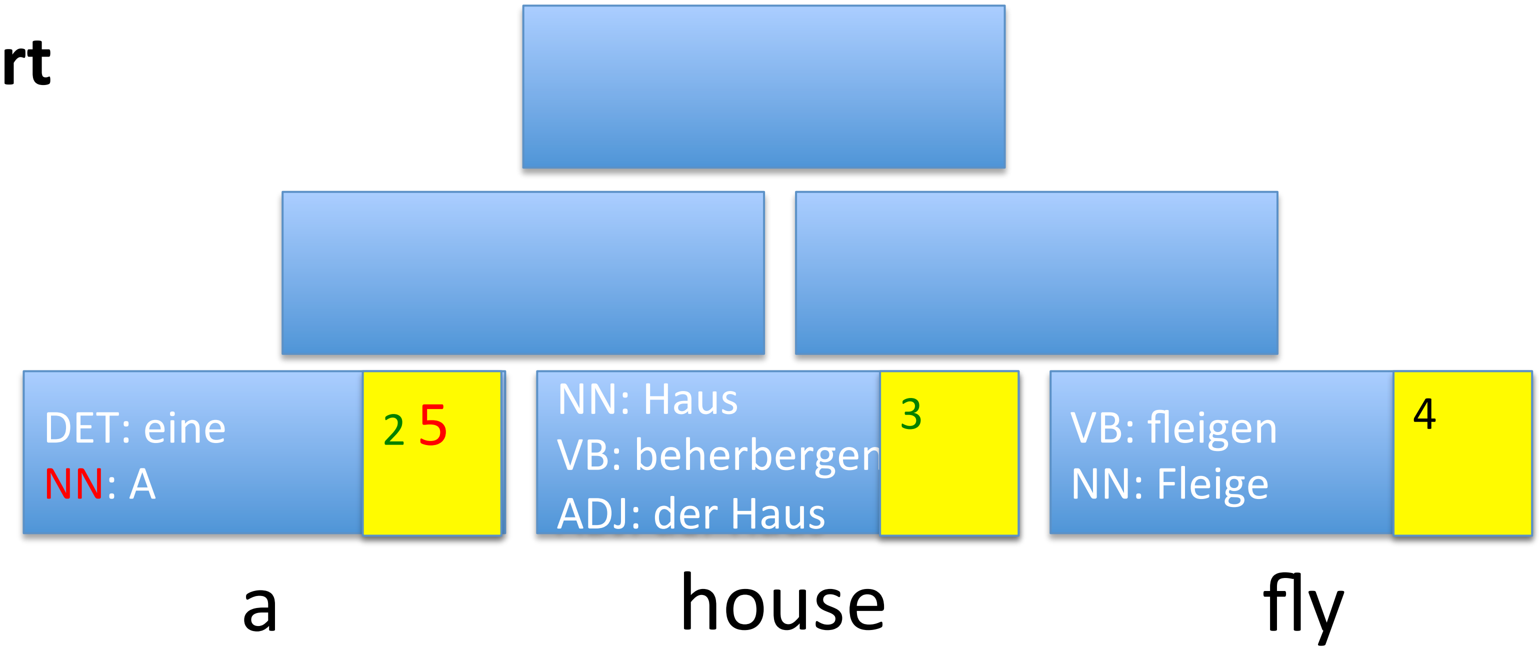
house

fly

Trie



Chart



DET: eine
NN: A

2 5

NN: Haus
VB: beherbergen
ADJ: der Haus

3

VB: fleigen
NN: Fleige

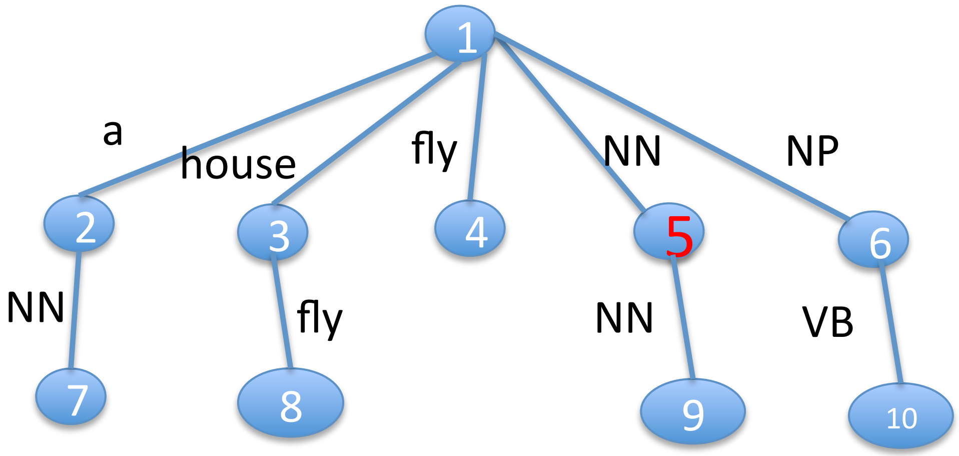
4

a

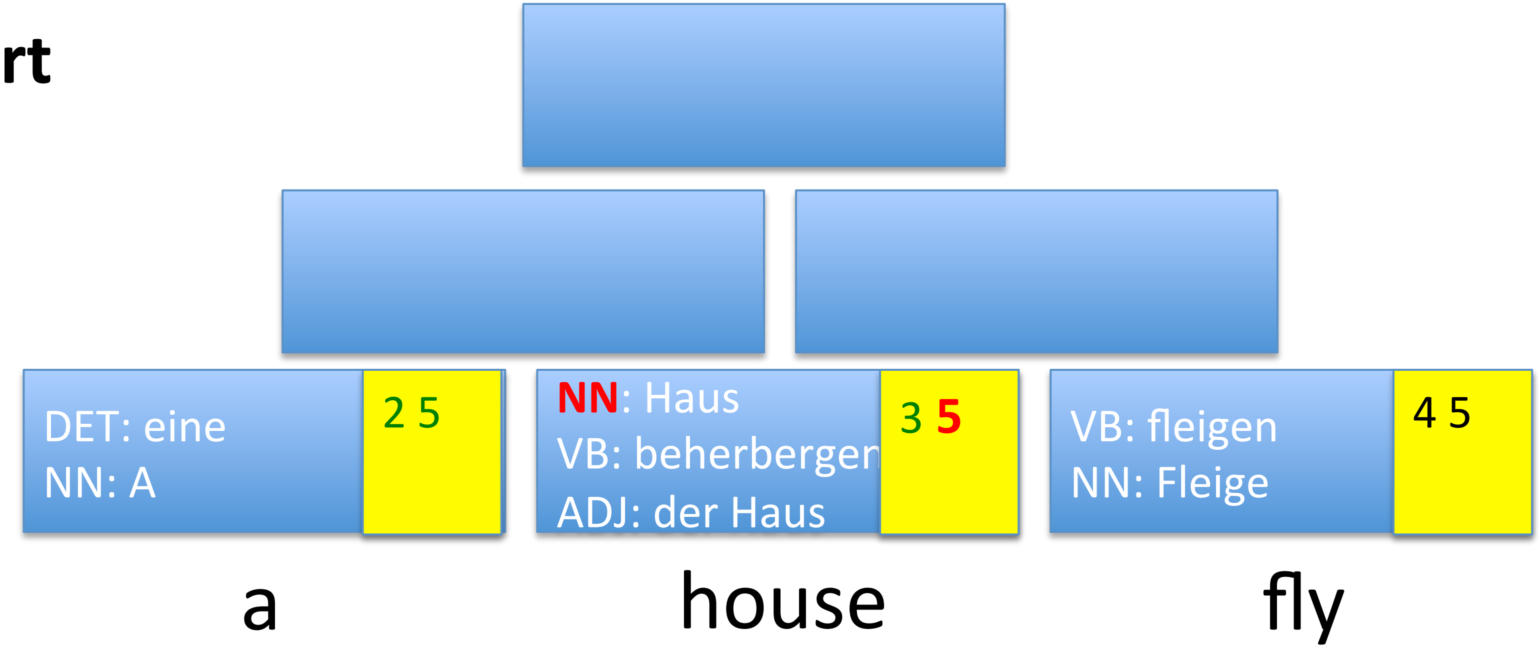
house

fly

Trie



Chart


DET: eine
NN: A

2 5

NN: Haus
VB: beherbergen
ADJ: der Haus

3 5

VB: fleigen
NN: Fleige

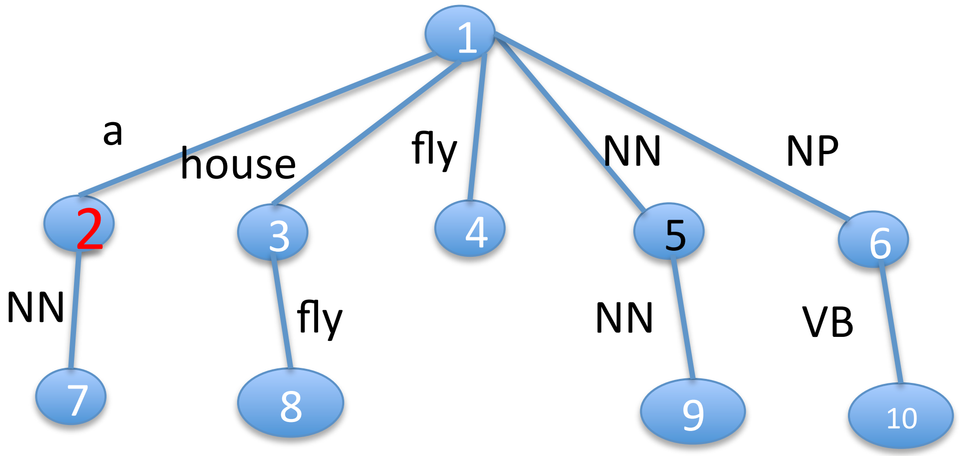
4 5

a

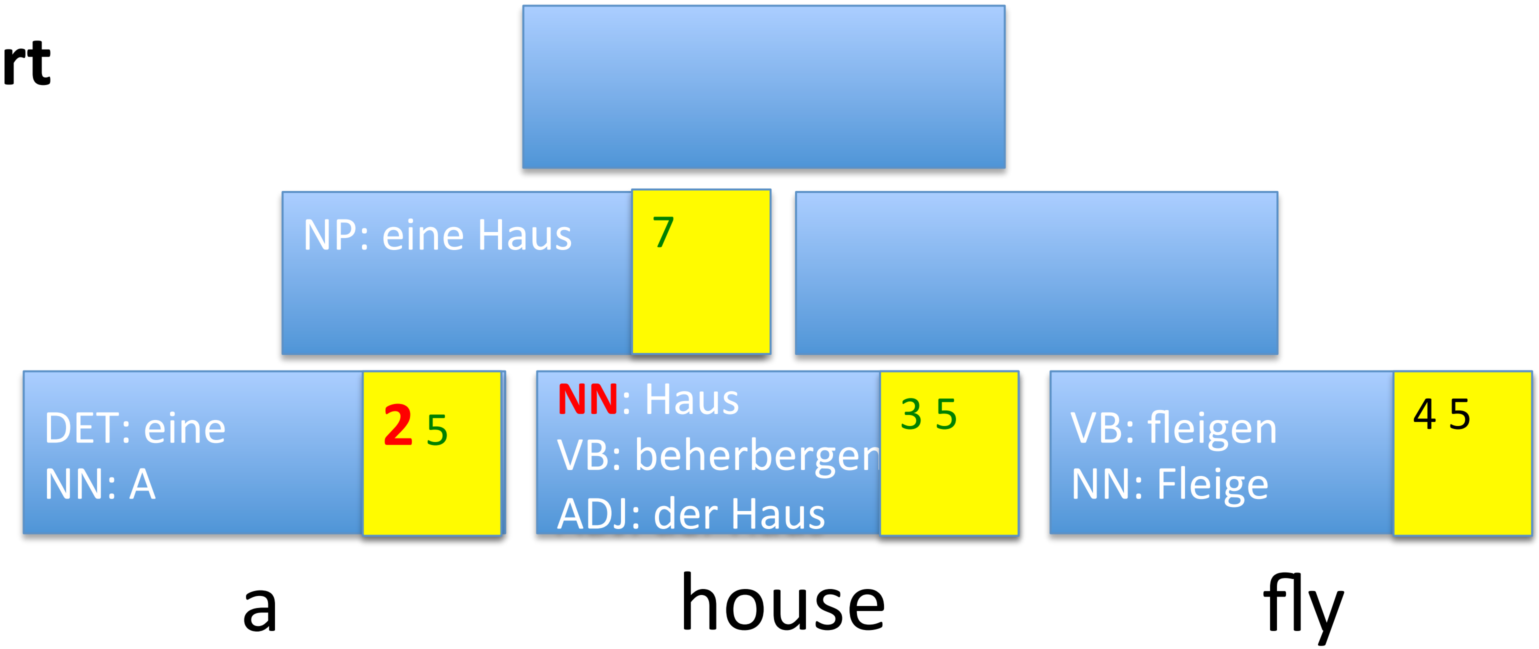
house

fly

Trie



Chart



NP: eine Haus

7

DET: eine
NN: A

2 5

NN: Haus
VB: beherbergen
ADJ: der Haus

3 5

VB: fleigen
NN: Fleige

4 5

a

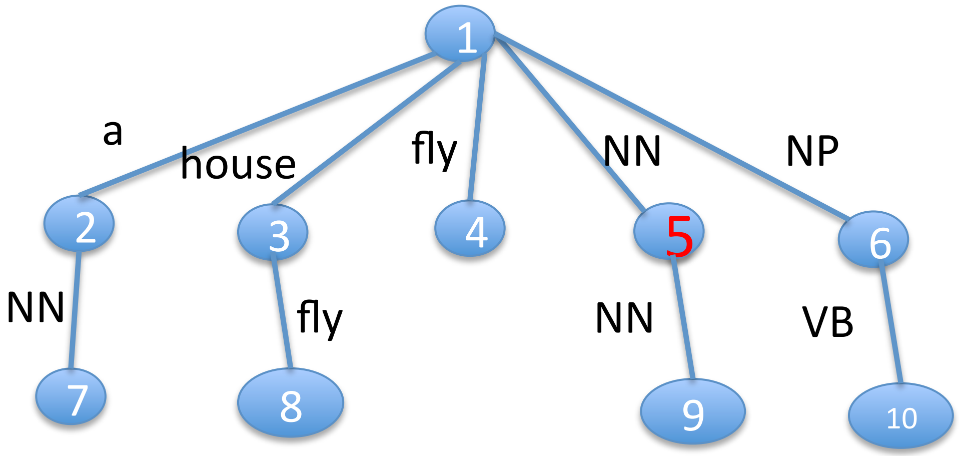
house

fly

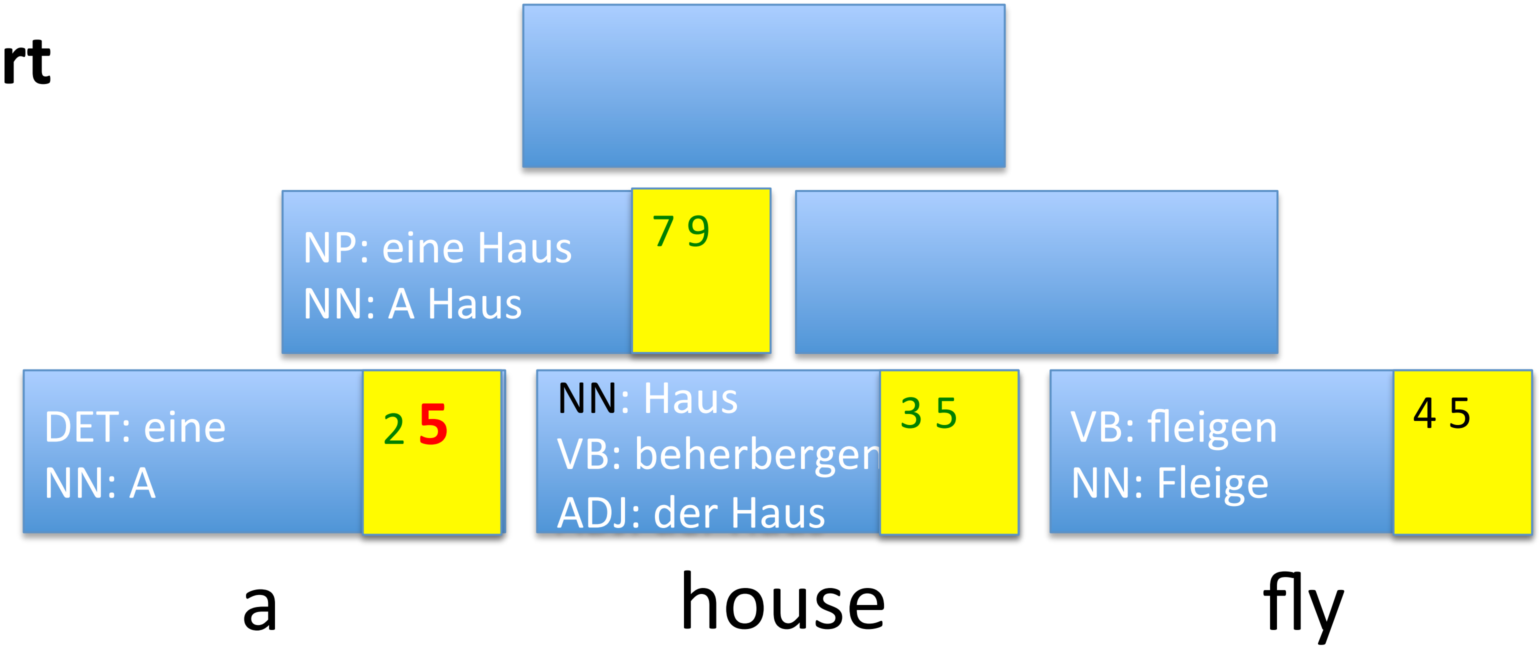
Do Extend node 2 with house, NN, VB, ADJ

Find NP → a NN$_1$ # eine NN$_1$

Trie



Chart



NP: eine Haus
NN: A Haus

7 9

DET: eine
NN: A

2 5

NN: Haus
VB: beherbergen
ADJ: der Haus

3 5

VB: fleigen
NN: Fleige

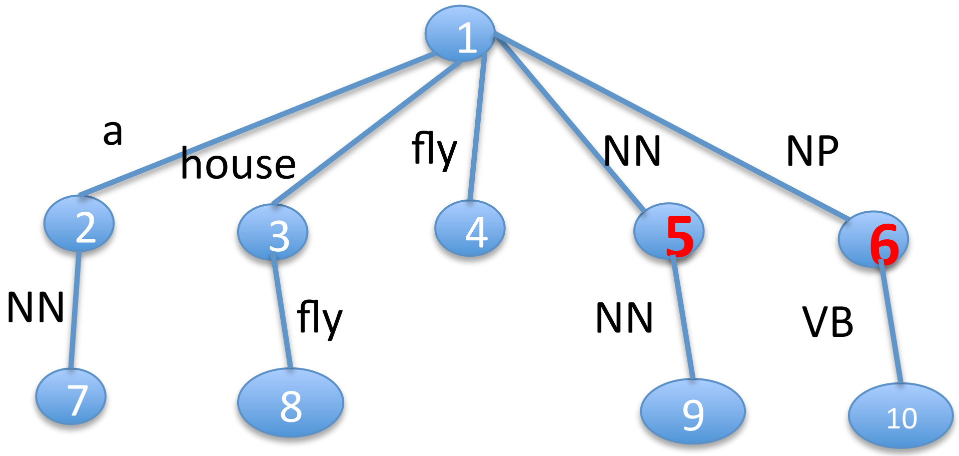
4 5

a house fly

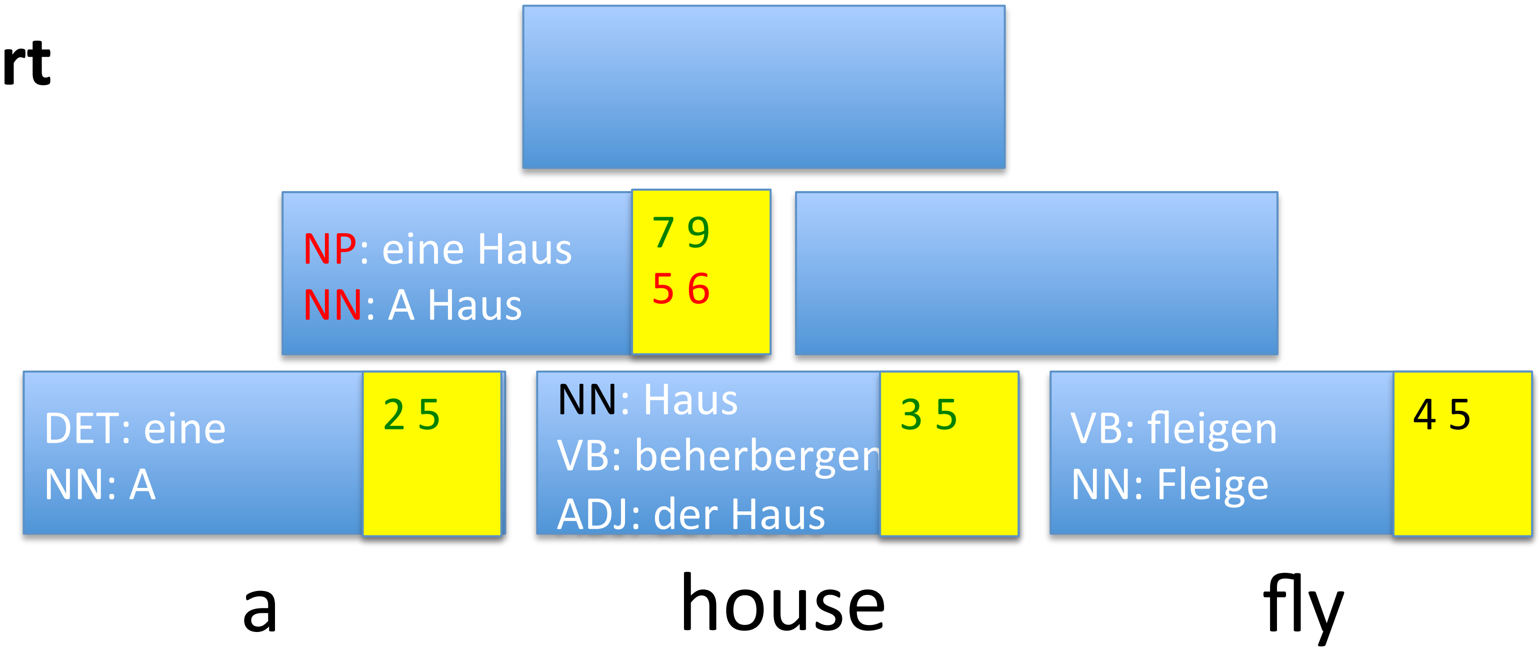
Do Extend node 5 with house, NN, VB, ADJ

Find $\text{NN} \rightarrow \text{NN}_1\ \text{NN}_2\ \#\ \text{NN}_1\ \text{NN}_2$

Trie



Chart



NP: eine Haus
NN: A Haus

7 9
5 6

DET: eine
NN: A

2 5

NN: Haus
VB: beherbergen
ADJ: der Haus

3 5

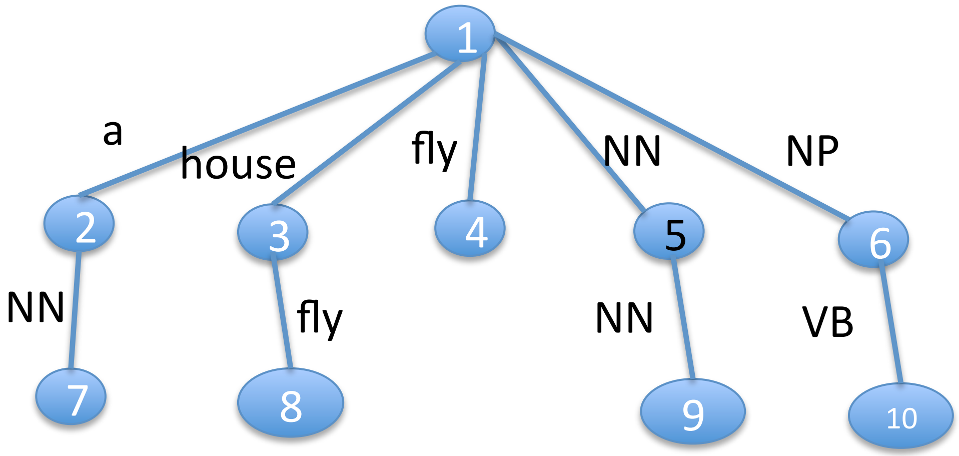
VB: fleigen
NN: Fleige

4 5

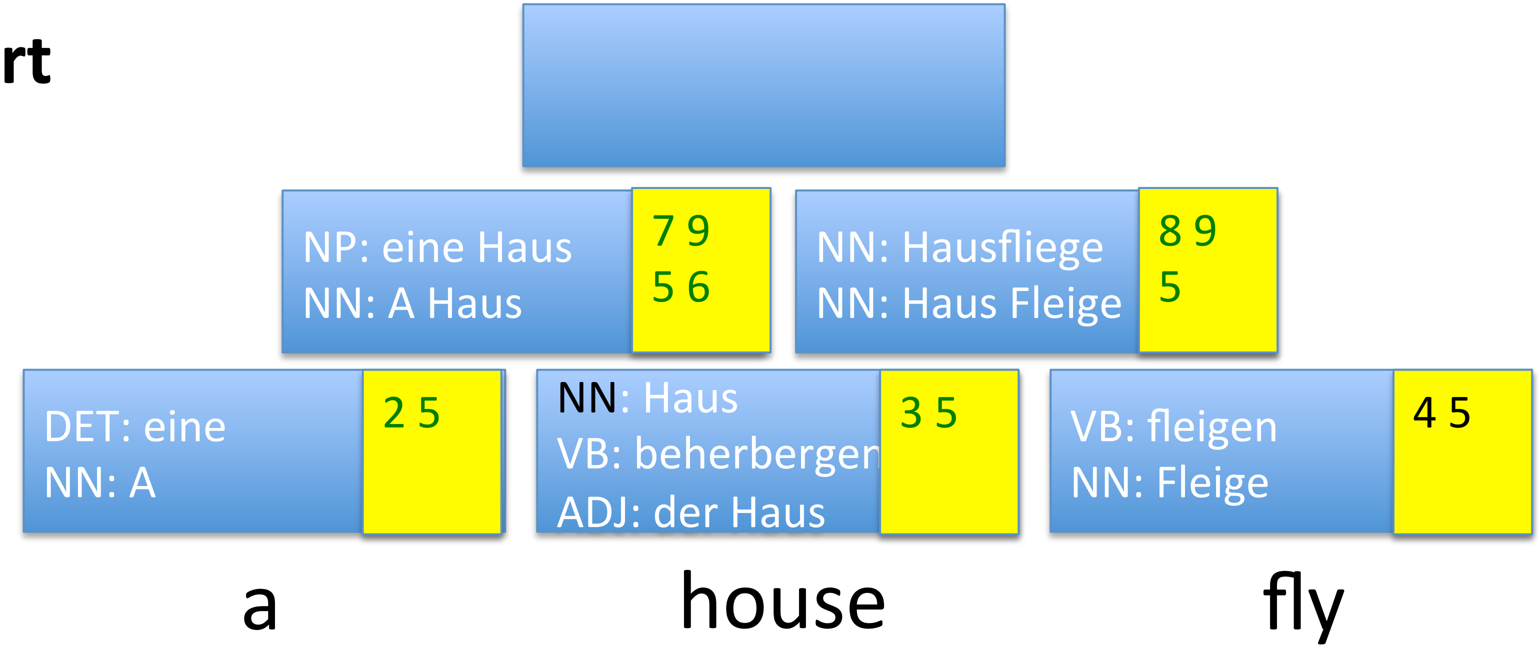
a house fly

Do Add nodes for NP and NN to active chart

Trie



Chart



NP: eine Haus
NN: A Haus
7 9
5 6

NN: Hausfliege
NN: Haus Fleige
8 9
5

DET: eine
NN: A
2 5

NN: Haus
VB: beherbergen
ADJ: der Haus
3 5

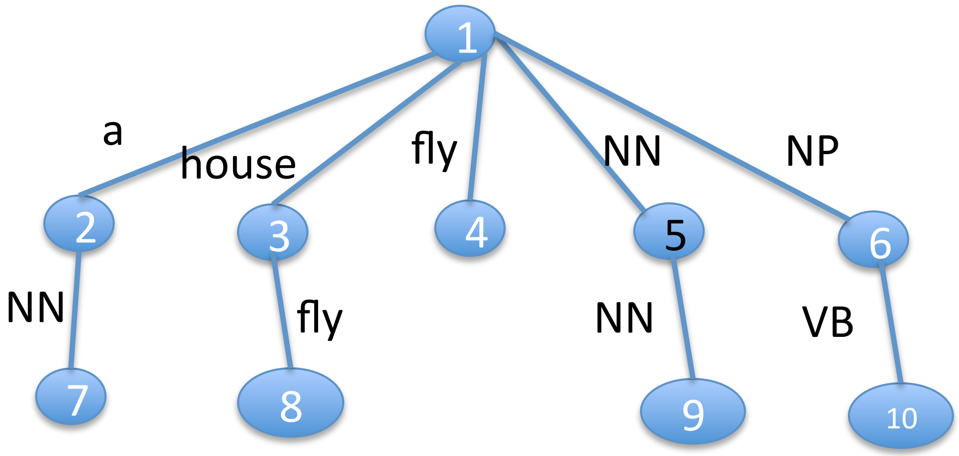
VB: fleigen
NN: Fleige
4 5

a

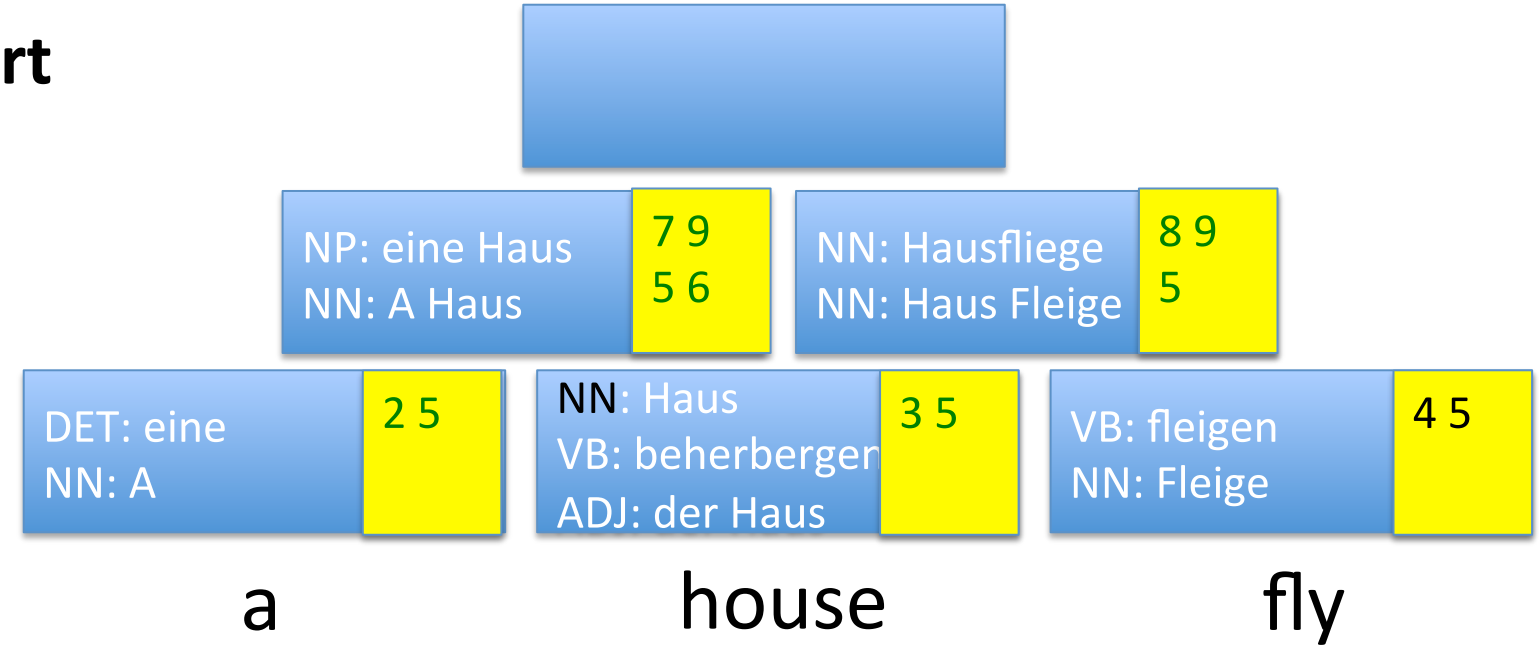
house

fly

Trie



Chart



NP: eine Haus
NN: A Haus
7 9
5 6

NN: Hausfliege
NN: Haus Fleige
8 9
5

DET: eine
NN: A
2 5

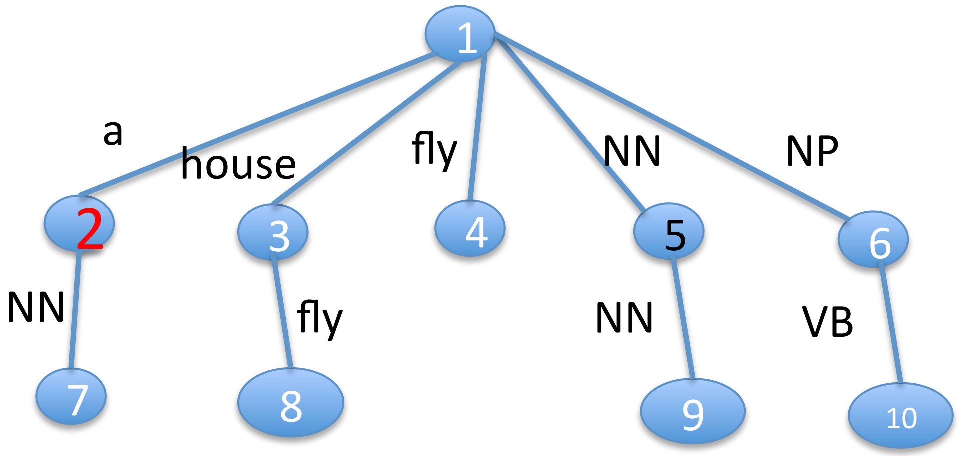
NN: Haus
VB: beherbergen
ADJ: der Haus
3 5

VB: fleigen
NN: Fleige
4 5

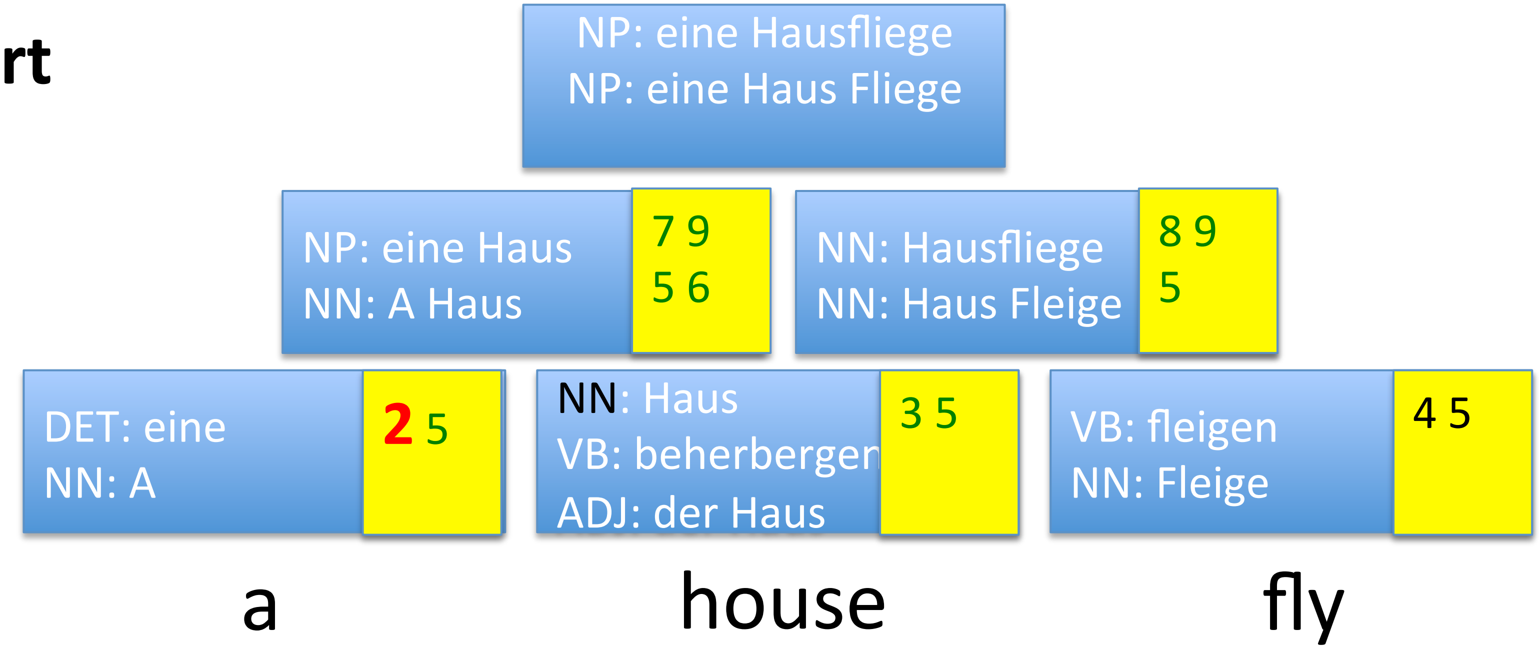
a house fly

Do Extend active chart in ‘a’ stack with non-terminals in ‘house fly’

Trie



Chart



NP: eine Hausfliege
NP: eine Haus Fliege

NP: eine Haus
NN: A Haus
7 9
5 6

NN: Hausfliege
NN: Haus Fleige
8 9
5

DET: eine
NN: A
2 5

NN: Haus
VB: beherbergen
ADJ: der Haus
3 5

VB: fleigen
NN: Fleige
4 5

a

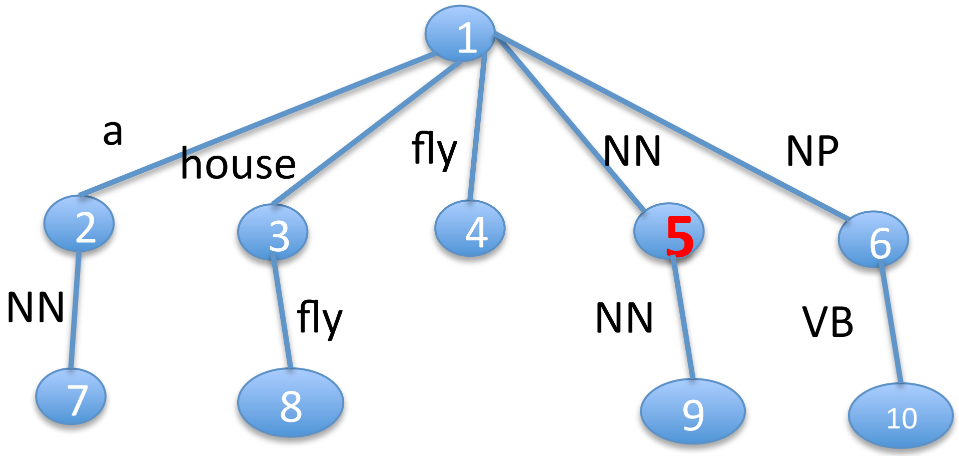
house

fly

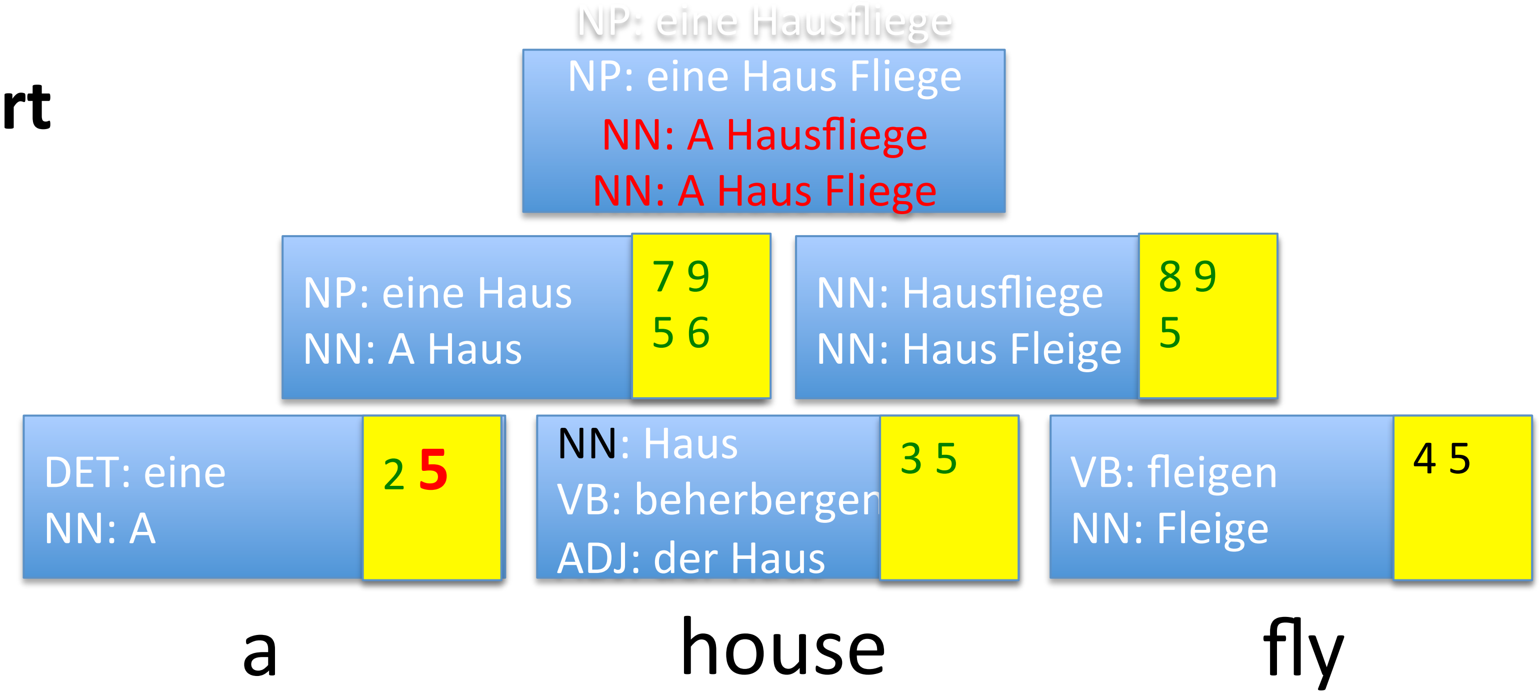
Find

NP → a NN_1 # eine NN_1

Trie



Chart



NP: eine Hausfliege

NP: eine Haus Fliege
NN: A Hausfliege
NN: A Haus Fliege

NP: eine Haus
NN: A Haus
7 9
5 6

NN: Hausfliege
NN: Haus Fleige
8 9
5

DET: eine
NN: A
2 5

NN: Haus
VB: beherbergen
ADJ: der Haus
3 5

VB: fleigen
NN: Fleige
4 5

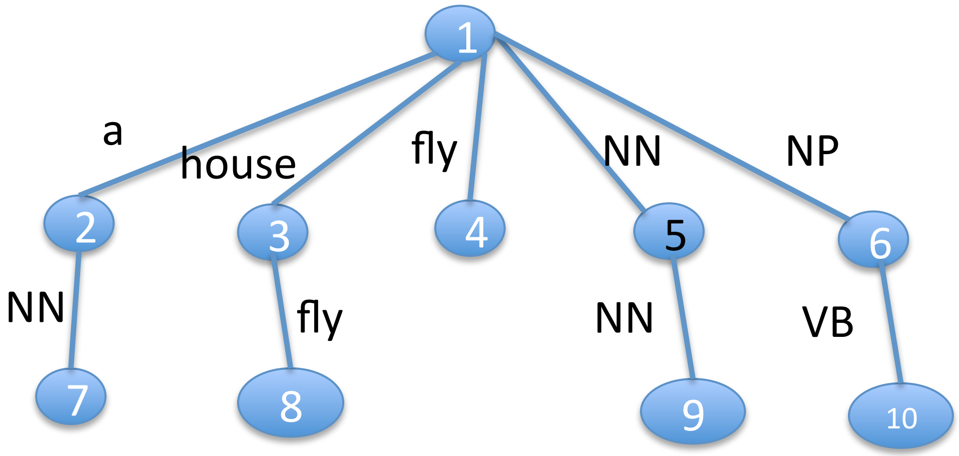
a

house

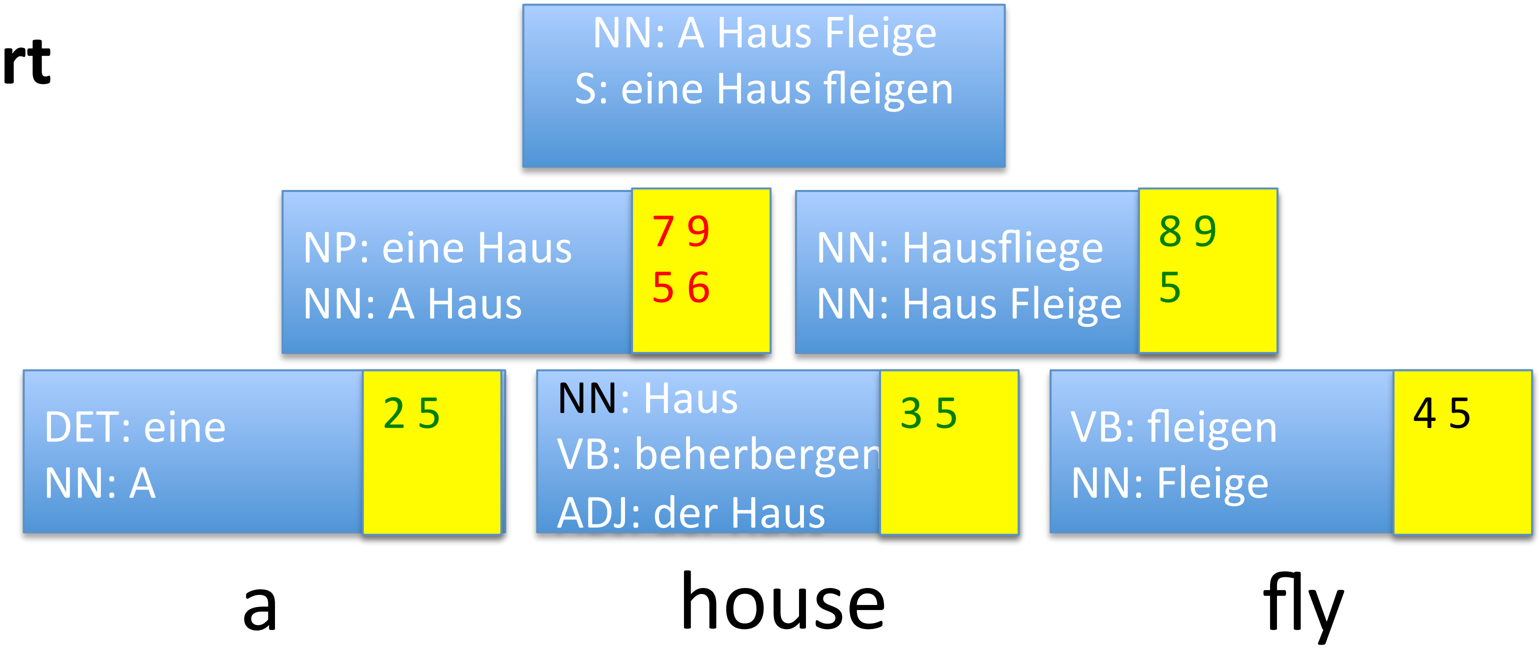
fly

Find NN → NN_1 NN_2 # NN_1 NN_2

Trie



Chart


NN: A Haus Fleige
S: eine Haus fleigen

NP: eine Haus
NN: A Haus

7 9
5 6

NN: Hausfliege
NN: Haus Fleige

8 9
5

DET: eine
NN: A

2 5

NN: Haus
VB: beherbergen
ADJ: der Haus

3 5

VB: fleigen
NN: Fleige

4 5

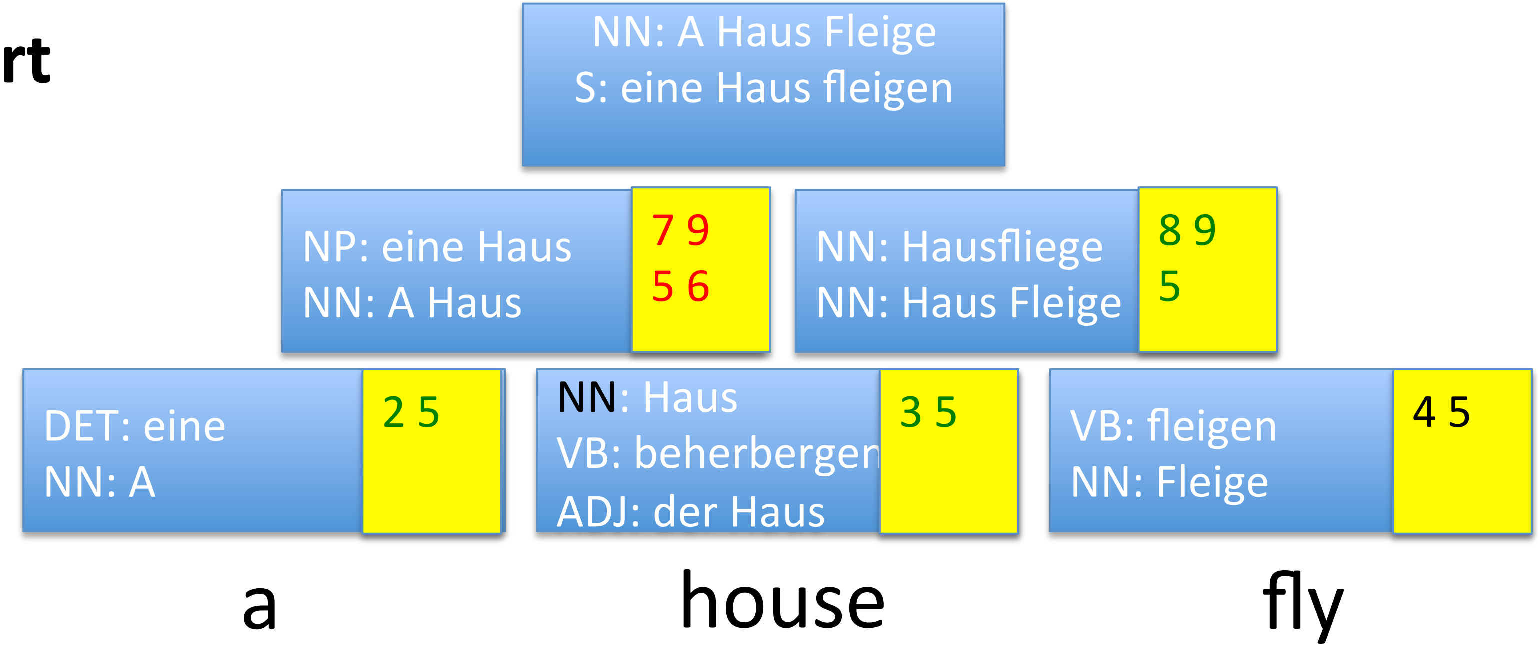
a

house

fly

Do Extend active chart in ‘a house’ stack with terminal AND non-terminals in ‘fly’

Chart



NN: A Haus Fleige
S: eine Haus fleigen

NP: eine Haus
NN: A Haus
7 9
5 6

NN: Hausfliege
NN: Haus Fleige
8 9
5

DET: eine
NN: A
2 5

NN: Haus
VB: beherbergen
ADJ: der Haus
3 5

VB: fleigen
NN: Fleige
4 5

a

house

fly

Number of lookups for top stack

= (size active stack ‘a’) * (number of NT in stack ‘house fly’)
+ (size active stack ‘a house’) * (number of NT & T in stack ‘fly’)
= (2 * 1) + (4 * 3)
= 14

Brute force method

= (3 * 4 * 3) + (2 * 3) + (3 * 1) = 45

CYK+

- Time Complexity

  $O(rn^3)$

  - n = length of input sentence
  - r = size of active chart
- Speedups
  - Reduce n
  - Maximum span of each rule
    - 10-20 words maximum
    - Similar to max reordering constraint

CYK+

- Reduce r (size of active chart)

  Size of r = O(C^R)

  - C = number of non-terminal labels
  - R = rank of grammar (max number of non-terminal per rule)
- Reduce C
  - Syntax → Hierarchical model
- Rules with consecutive source non-terminals

  X → gibt X_1 X_2 #   gives X_1 to X_2