

# Der CKY-Parser

Vorlesung “Computerlinguistische Techniken”  
Alexander Koller

27. Oktober 2015

# Übersicht

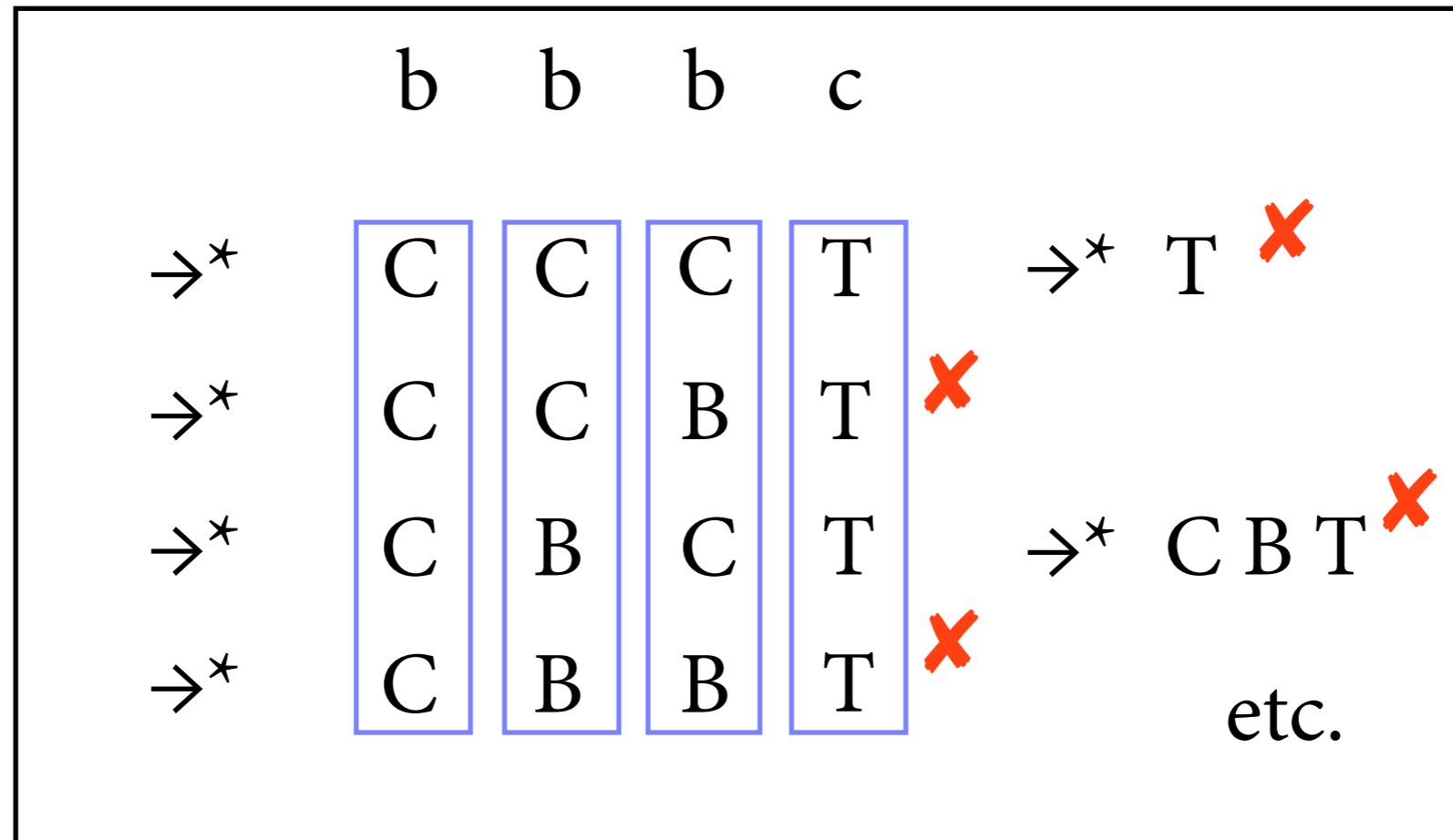
- Der CKY-Parser
- Komplexität des CKY-Algorithmus
- Implementierung in Python

# Shift-Reduce-Parsing

- Shift-Regel:  
 $(a \cdot w, s) \rightarrow (w, s \cdot a)$
- Reduce-Regel:  
 $(w, s \cdot w') \rightarrow (w, s \cdot A)$  falls  $A \rightarrow w'$  in  $P$
- Start:  $(w, \varepsilon)$
- Wende Regeln nichtdeterministisch an,  
bis Zustand  $(\varepsilon, S)$  erreicht ist

# Shift-Reduce: Ein Problem

|                     |                   |                   |
|---------------------|-------------------|-------------------|
| $S \rightarrow B S$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow C T$ | $C \rightarrow b$ | $T \rightarrow c$ |



SR-Erkennen muss für String der Länge  $n$  bis zu  $2^n$  Kombinationen durchprobieren.

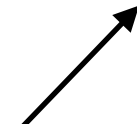
# Laufzeitvergleich


```
def quadratic_issorted(L):  
    for i in range(len(L)):  
        for j in range(i+1, len(L)):  
            if L[j] < L[i]:  
                return False  
    return True
```

```
def linear_issorted(L):  
    for i in range(len(L)-1):  
        if L[i] > L[i+1]:  
            return False  
    return True
```

## Laufzeit

| len(L)    | quadratic | linear  |
|-----------|-----------|---------|
| 100       | 0.5 ms    | 0.02 ms |
| 1000      | 40 ms     | 0.1 ms  |
| 10000     | 4.5 sec   | 1.2 ms  |
| 100.000   | 464 sec   | 13 ms   |
| 1.000.000 |           | 179 ms  |

$$\approx n^2 \cdot 45 \text{ ns}$$


$$\approx n \cdot 120 \text{ ns}$$


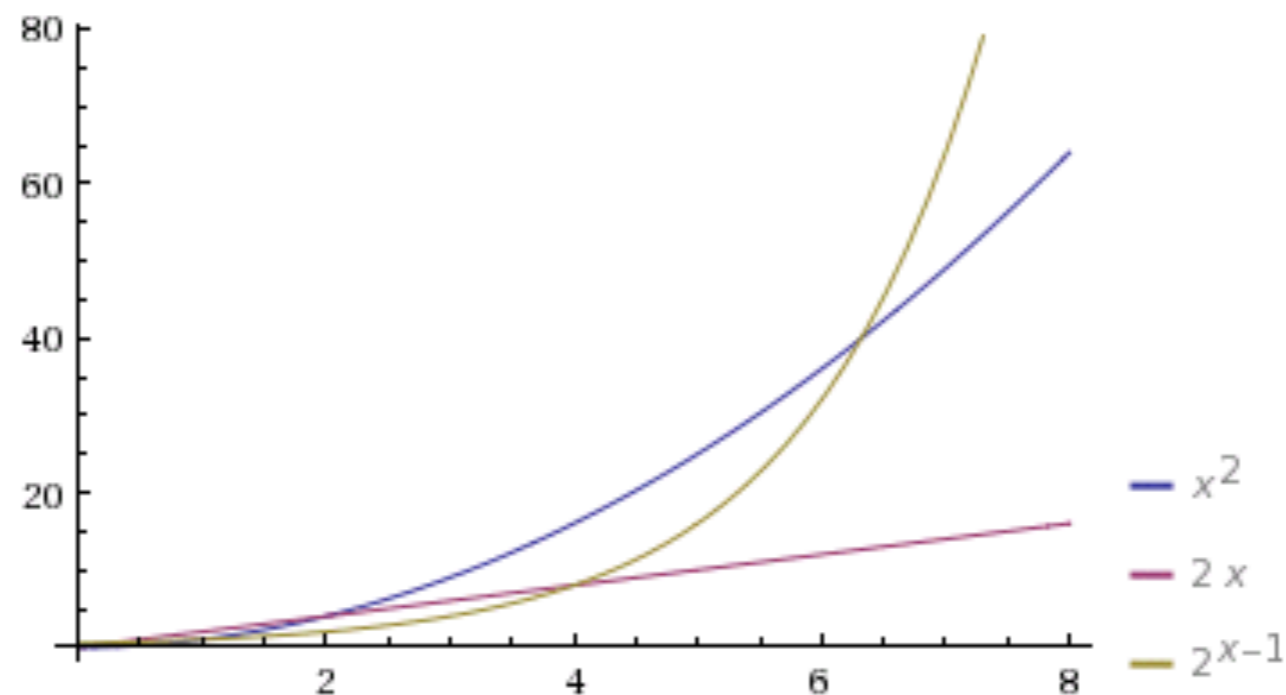
# O-Notation

- Asymptotische Laufzeit eines Algorithmus:  
Abstrahiert über Implementierungsdetails.
- Seien  $f, g$  Funktionen. Definition:

$$f = O(g) \text{ gdw.}$$
$$\text{ex. } c, n_0 \text{ mit } f(n) \leq c \cdot g(n) \text{ f.a. } n \geq n_0$$
- Man nimmt normalerweise kleinstes  $g$ ,  
für das  $f = O(g)$ .

# Laufzeitklassen

- Worst-Case-Laufzeit von Shift-Reduce:  
 $2^n$  Berechnungsschritte ( $n$  ist Länge des Strings).
- Exponentialfunktion wächst schneller als jedes Polynom: Es gibt kein  $k$ , so dass  $2^n = O(n^k)$ .



# Was ist das Problem?

- Warum braucht der SR-Parser exponentielle Laufzeit?
- Zwischenergebnisse werden mehrfach berechnet. Können wir das vermeiden?

|                 |   |   |   |   |
|-----------------|---|---|---|---|
|                 | b | b | b | c |
| $\rightarrow^*$ | C | C | C | T |
| $\rightarrow^*$ | C | C | B | T |
| $\rightarrow^*$ | C | B | C | T |
| $\rightarrow^*$ | C | B | B | T |



# Auswege

- Für top-down: *Memoisierung*. Speichere frühere berechnete Zwischenergebnisse in Tabelle und schlage sie beim zweiten Aufruf nach.
- Für bottom-up: *Dynamisches Programmieren* (auch bekannt als *Chart-Parsing*):

Algorithmus arbeitet direkt auf einer Tabelle (der Chart).

# Der CKY-Parser

- Einfachster Chartparser für kfGs in CNF.
- Erfunden in den 1960ern von Cocke, Younger, Kasami; heißt manchmal auch CYK-Parser.
- Berechnet bottom-up Aussagen der Form “ $A \Rightarrow^* w_i \dots w_{k-1} ?$ ”.

# Der CKY-Parser

|                                |                              |                                     |
|--------------------------------|------------------------------|-------------------------------------|
| $S \rightarrow NP VP$          | $V \rightarrow \text{isst}$  | $\text{Det} \rightarrow \text{ein}$ |
| $NP \rightarrow \text{Det } N$ | $NP \rightarrow \text{Hans}$ | $N \rightarrow \text{Käsebrod}$     |
| $VP \rightarrow V NP$          |                              |                                     |

Hans isst ein Käsebrod

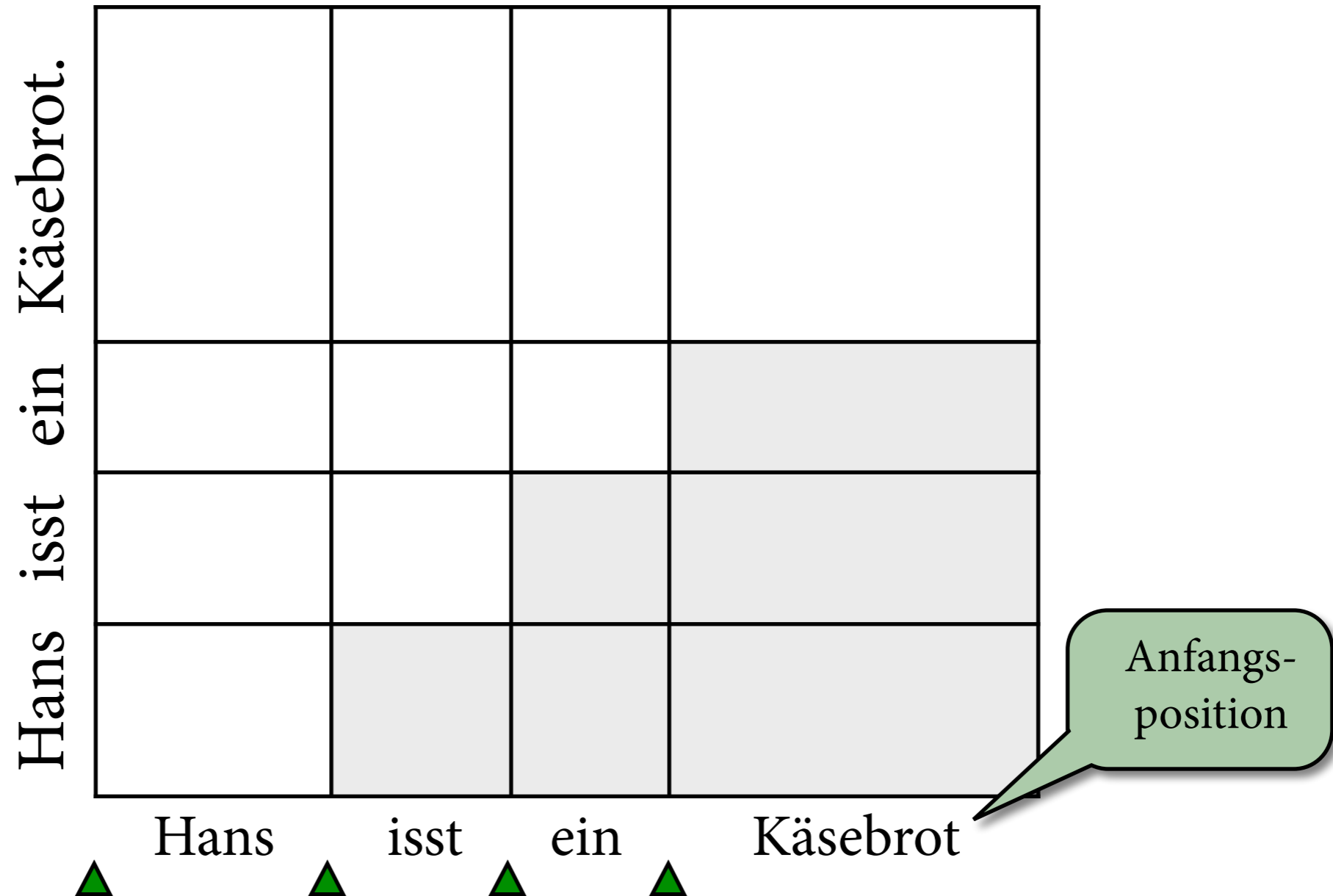
# Der CKY-Parser

|                                |                              |                                     |
|--------------------------------|------------------------------|-------------------------------------|
| $S \rightarrow NP VP$          | $V \rightarrow \text{isst}$  | $\text{Det} \rightarrow \text{ein}$ |
| $NP \rightarrow \text{Det } N$ | $NP \rightarrow \text{Hans}$ | $N \rightarrow \text{Käsebrod}$     |
| $VP \rightarrow V NP$          |                              |                                     |

|                         |      |      |     |          |
|-------------------------|------|------|-----|----------|
| Hans isst ein Käsebrod. |      |      |     |          |
|                         |      |      |     |          |
|                         |      |      |     |          |
|                         |      |      |     |          |
|                         | Hans | isst | ein | Käsebrod |

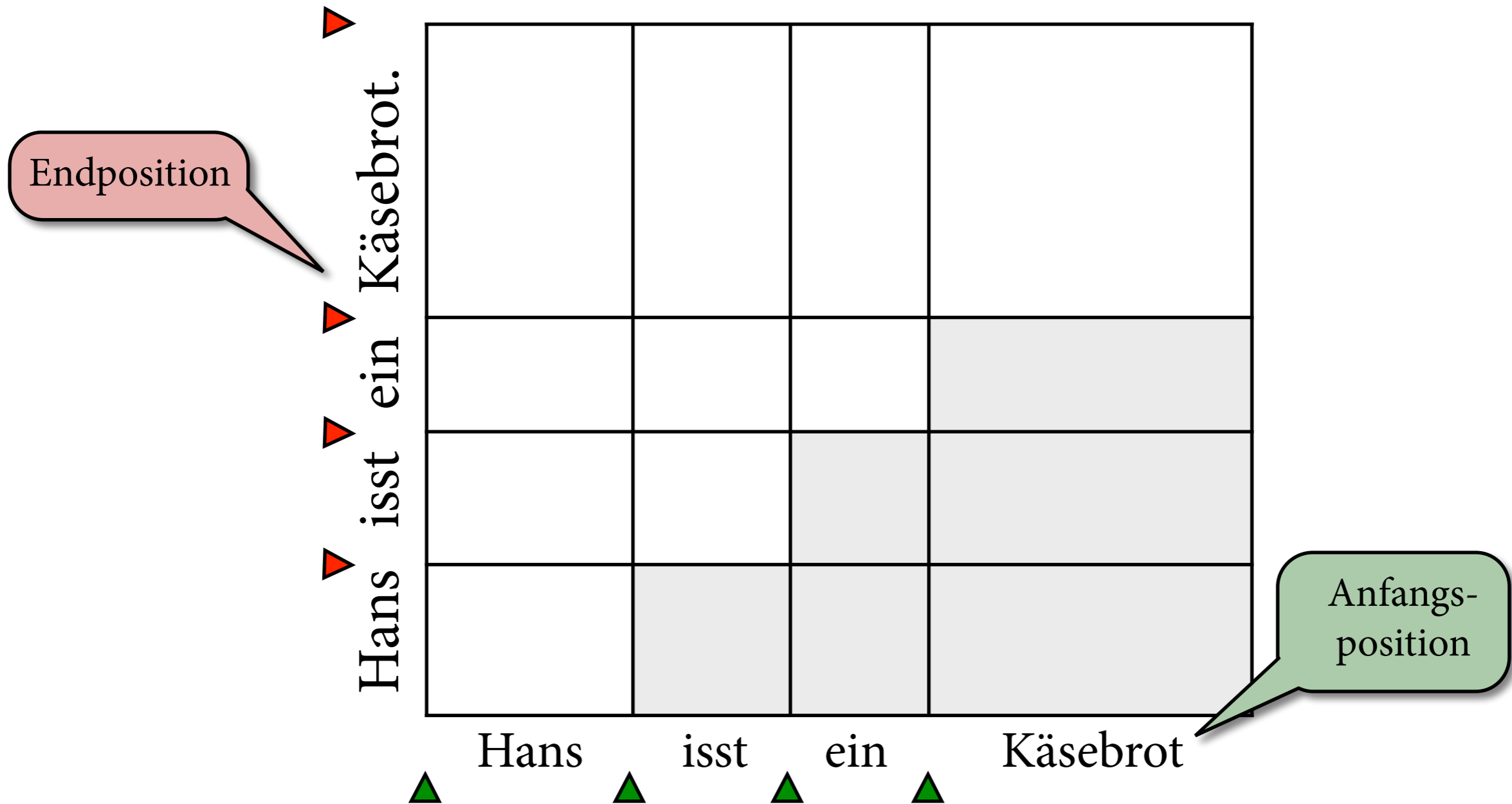
# Der CKY-Parser

|                                |                              |                                     |
|--------------------------------|------------------------------|-------------------------------------|
| $S \rightarrow NP VP$          | $V \rightarrow \text{isst}$  | $\text{Det} \rightarrow \text{ein}$ |
| $NP \rightarrow \text{Det } N$ | $NP \rightarrow \text{Hans}$ | $N \rightarrow \text{Käsebrod}$     |
| $VP \rightarrow V NP$          |                              |                                     |



# Der CKY-Parser

|                                |                              |                                     |
|--------------------------------|------------------------------|-------------------------------------|
| $S \rightarrow NP VP$          | $V \rightarrow \text{isst}$  | $\text{Det} \rightarrow \text{ein}$ |
| $NP \rightarrow \text{Det } N$ | $NP \rightarrow \text{Hans}$ | $N \rightarrow \text{Käsebrod}$     |
| $VP \rightarrow V NP$          |                              |                                     |

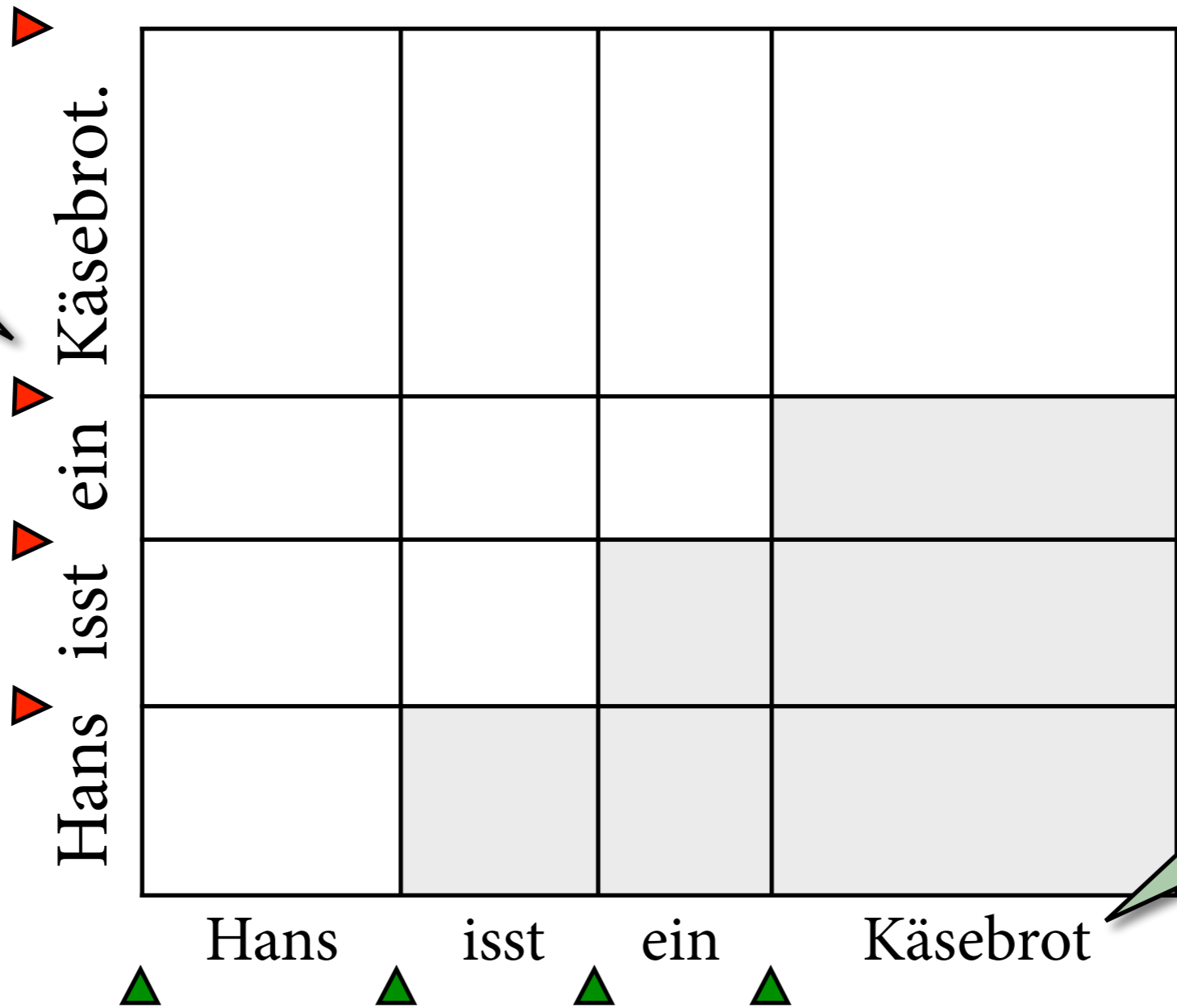


# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow \text{isst}$        $\text{Det} \rightarrow \text{ein}$   
 $NP \rightarrow \text{Det } N$        $NP \rightarrow \text{Hans}$        $N \rightarrow \text{Käsebrod}$   
 $VP \rightarrow V NP$

Chart

Endposition



Anfangsposition

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow \text{isst}$        $\text{Det} \rightarrow \text{ein}$   
 $NP \rightarrow \text{Det } N$        $NP \rightarrow \text{Hans}$        $N \rightarrow \text{Käsebrod}$   
 $VP \rightarrow V NP$

Chart

Endposition

|   |      |      |     |          |
|---|------|------|-----|----------|
| ▼ |      |      |     |          |
| ▼ |      |      |     |          |
| ▼ |      |      |     |          |
| ▼ | NP   |      |     |          |
|   | Hans | isst | ein | Käsebrod |
|   | ▲    | ▲    | ▲   | ▲        |

Anfangsposition



# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow isst$        $Det \rightarrow ein$   
 $NP \rightarrow Det N$        $NP \rightarrow Hans$        $N \rightarrow Käsebro$   
 $VP \rightarrow V NP$

Chart

Endposition

A CKY parse chart for the sentence "Hans isst ein Käsebrot.". The chart is a 4x4 grid. The columns are labeled with the words "Hans", "isst", "ein", and "Käsebrot". The rows are labeled with the words "Hans", "isst", "ein", and "Käsebrot". The cell at row 1, column 1 contains "NP". The cell at row 2, column 2 contains "V". The cell at row 3, column 3 contains "ein". The cell at row 4, column 4 contains "Käsebrot". The cells at (1,2), (1,3), (1,4), (2,3), (2,4), (3,4), and (4,1) are shaded gray. Red triangles point to the top of each row, and green triangles point to the bottom of each column.

|      |      |     |          |
|------|------|-----|----------|
|      |      |     |          |
|      |      |     |          |
|      | V    |     |          |
| NP   |      |     |          |
| Hans | isst | ein | Käsebrot |

Anfangsposition

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow \text{isst}$        $\text{Det} \rightarrow \text{ein}$   
 $NP \rightarrow \text{Det } N$        $NP \rightarrow \text{Hans}$        $N \rightarrow \text{Käse}$   
 $VP \rightarrow V NP$        $N \rightarrow \text{brot}$

Chart

Endposition

Hans isst ein Käse**.**

|      |      |     |               |
|------|------|-----|---------------|
|      |      |     |               |
|      |      | Det |               |
|      | V    |     |               |
| NP   |      |     |               |
| Hans | isst | ein | Käse <b>.</b> |

▲      ▲      ▲      ▲

Anfangs-  
position

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow isst$        $Det \rightarrow ein$   
 $NP \rightarrow Det N$        $NP \rightarrow Hans$        $N \rightarrow Käsebro$   
 $VP \rightarrow V NP$

Chart

Endposition

Hans isst ein Käsebro.

|      |      |     |         |
|------|------|-----|---------|
|      |      |     | N       |
|      |      | Det |         |
|      | V    |     |         |
| NP   |      |     |         |
| Hans | isst | ein | Käsebro |

Anfangsposition

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow \text{isst}$        $\text{Det} \rightarrow \text{ein}$   
 $NP \rightarrow \text{Det } N$        $NP \rightarrow \text{Hans}$        $N \rightarrow \text{Käse}$   
 $VP \rightarrow V NP$        $N \rightarrow \text{brot}$

Chart

Endposition

|      |      |     |      |      |
|------|------|-----|------|------|
|      |      |     |      |      |
|      |      | NP  | N    |      |
|      |      | Det |      |      |
|      | V    |     |      |      |
| NP   |      |     |      |      |
| Hans | isst | ein | Käse | brot |

Hans isst ein Käse**.**

Anfangsposition

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow isst$        $Det \rightarrow ein$   
 $NP \rightarrow Det N$        $NP \rightarrow Hans$        $N \rightarrow Käsebro$   
 $VP \rightarrow V NP$

Chart

Endposition

|      |      |     |         |
|------|------|-----|---------|
|      |      |     |         |
|      |      | NP  | N       |
|      |      | Det |         |
|      | V    |     |         |
| NP   |      |     |         |
| Hans | isst | ein | Käsebro |

Anfangsposition

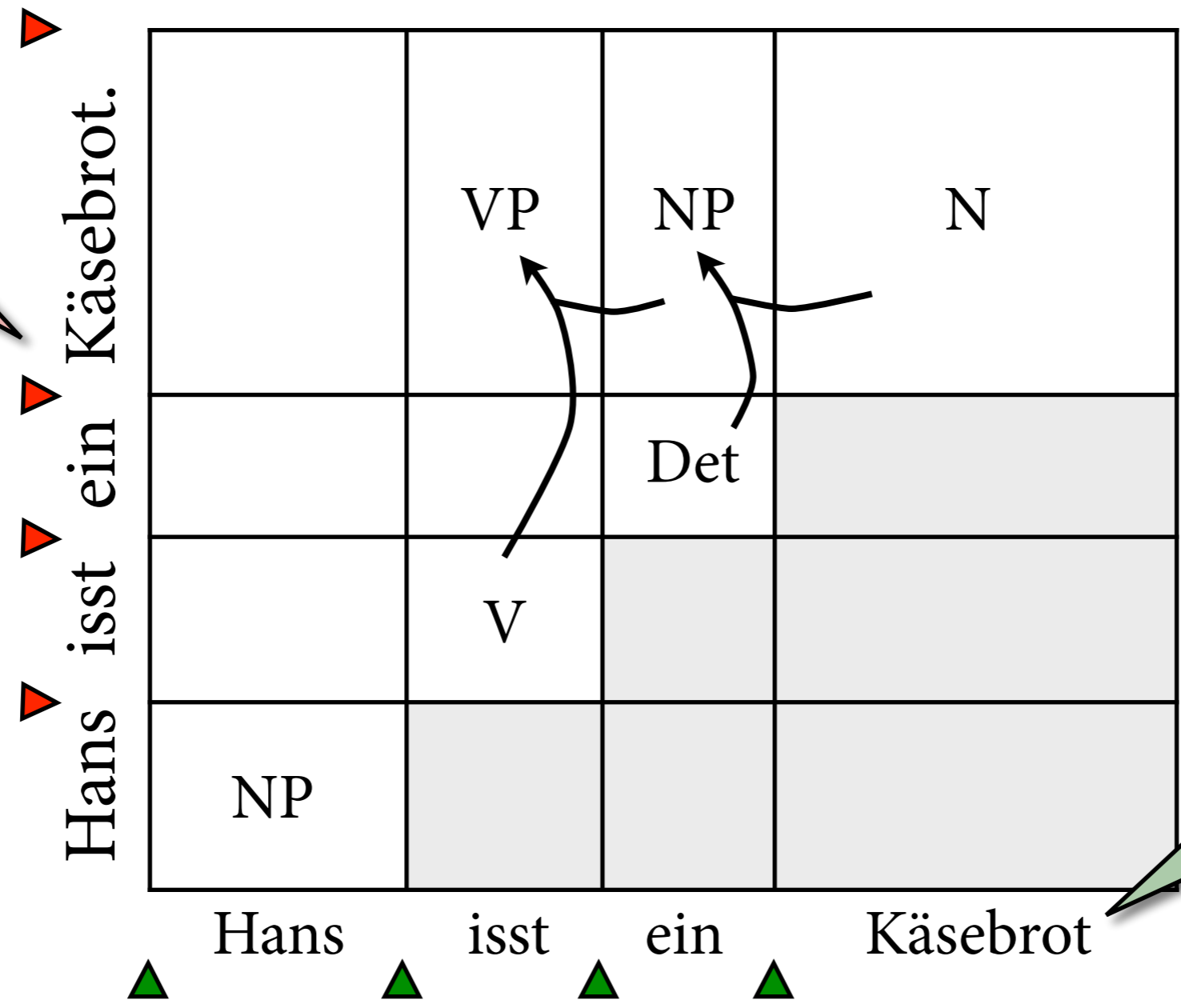
Hans isst ein Käsebro.

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow isst$        $Det \rightarrow ein$   
 $NP \rightarrow Det N$        $NP \rightarrow Hans$        $N \rightarrow Käsebro$   
 $VP \rightarrow V NP$

Chart

Endposition



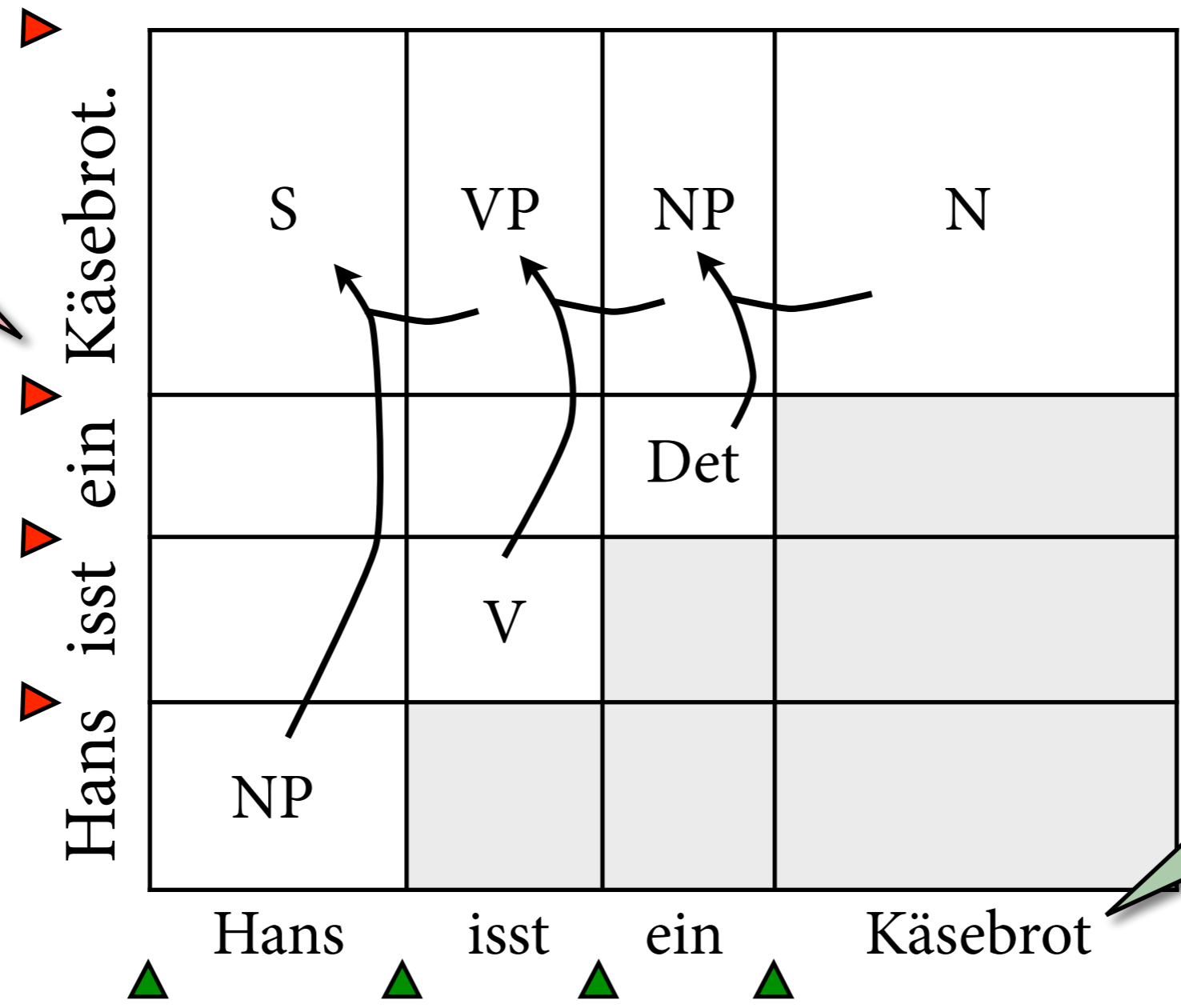
Anfangsposition

# Der CKY-Parser

$S \rightarrow NP VP$        $V \rightarrow isst$        $Det \rightarrow ein$   
 $NP \rightarrow Det N$        $NP \rightarrow Hans$        $N \rightarrow Käsebro$   
 $VP \rightarrow V NP$

Chart

Endposition

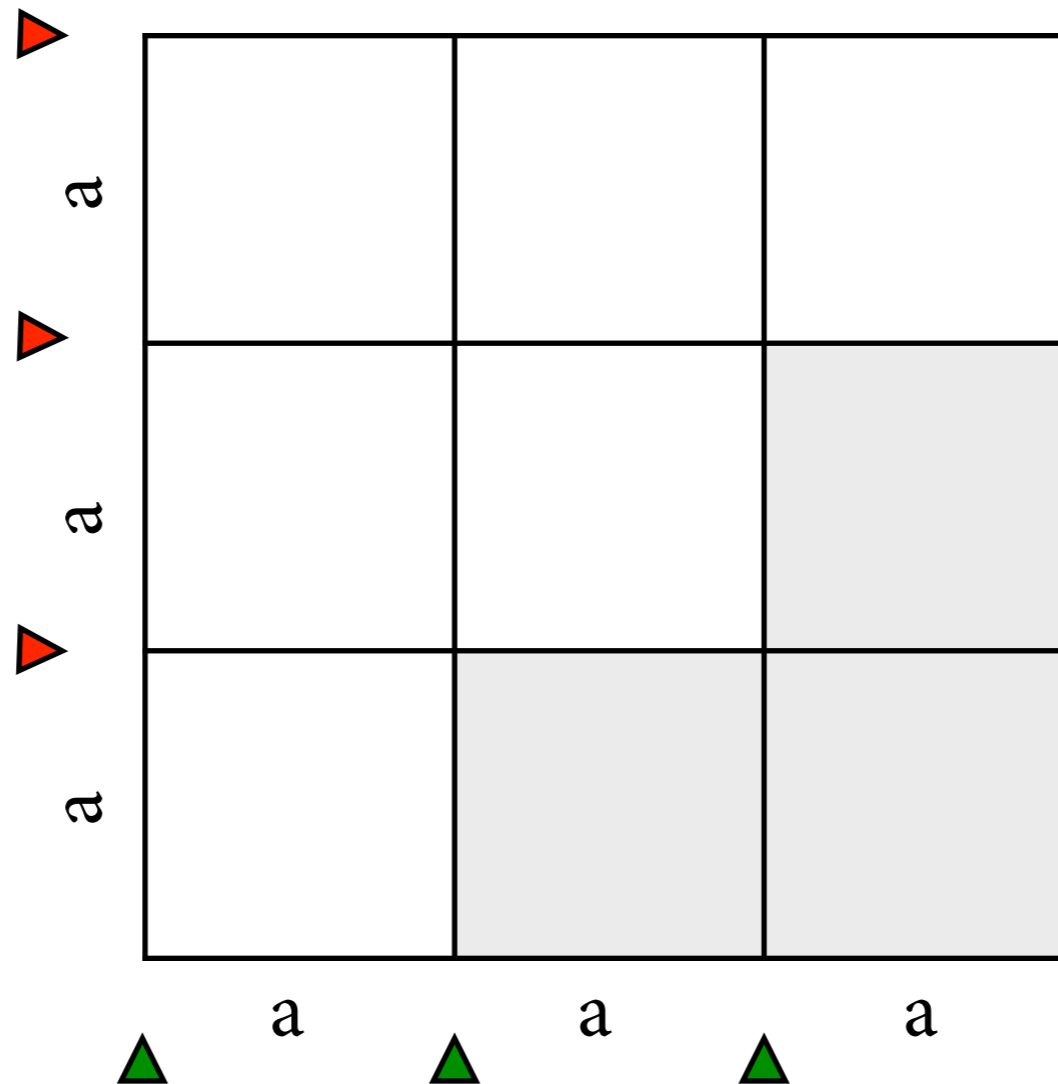


Anfangsposition

# Der CKY-Parser

$S \rightarrow SS$

$S \rightarrow a$

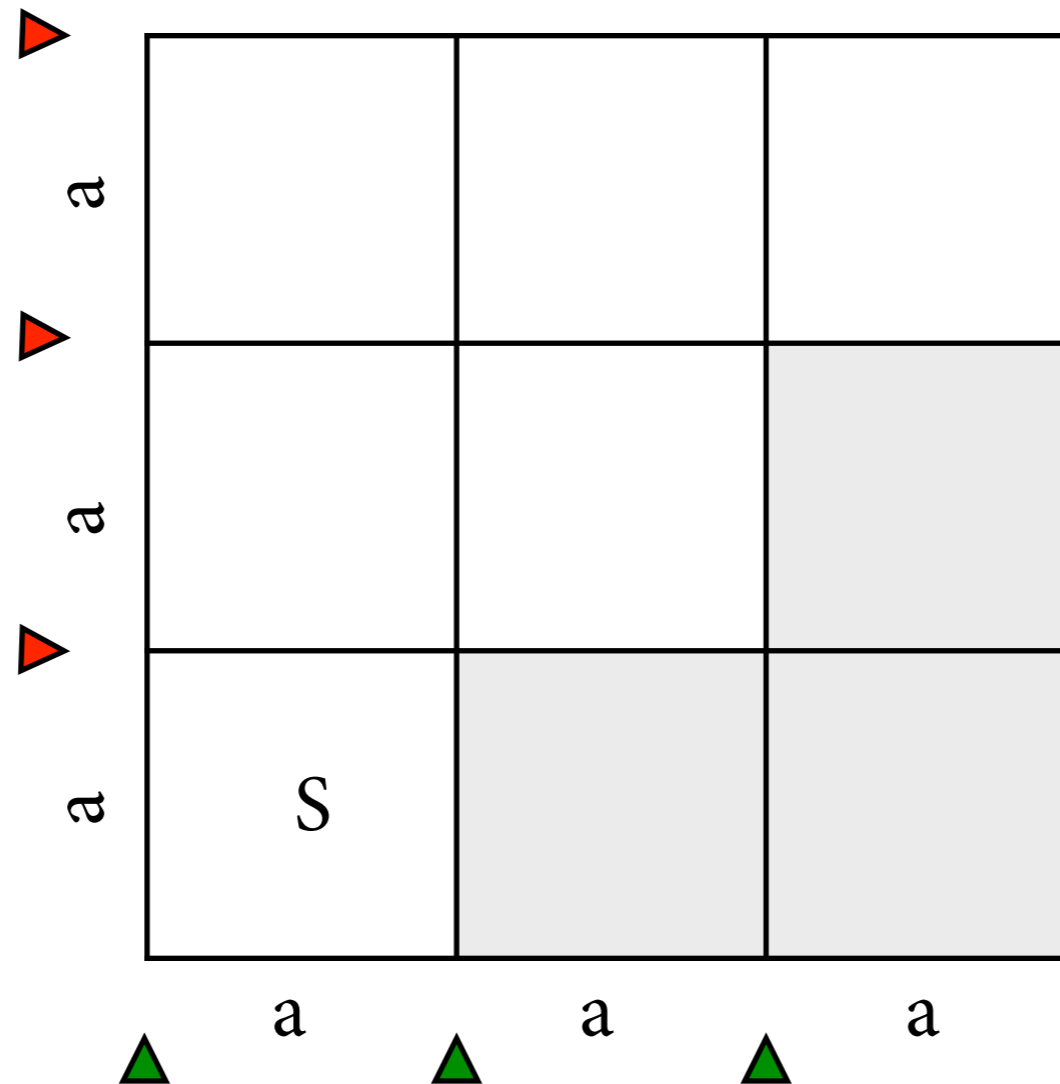




# Der CKY-Parser

$S \rightarrow SS$

$S \rightarrow a$



# Der CKY-Parser

$S \rightarrow SS$

$S \rightarrow a$

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
| a |   |   |   |
| a | S |   |   |
| a | S |   |   |
|   | a | a | a |

# Der CKY-Parser

$S \rightarrow SS$

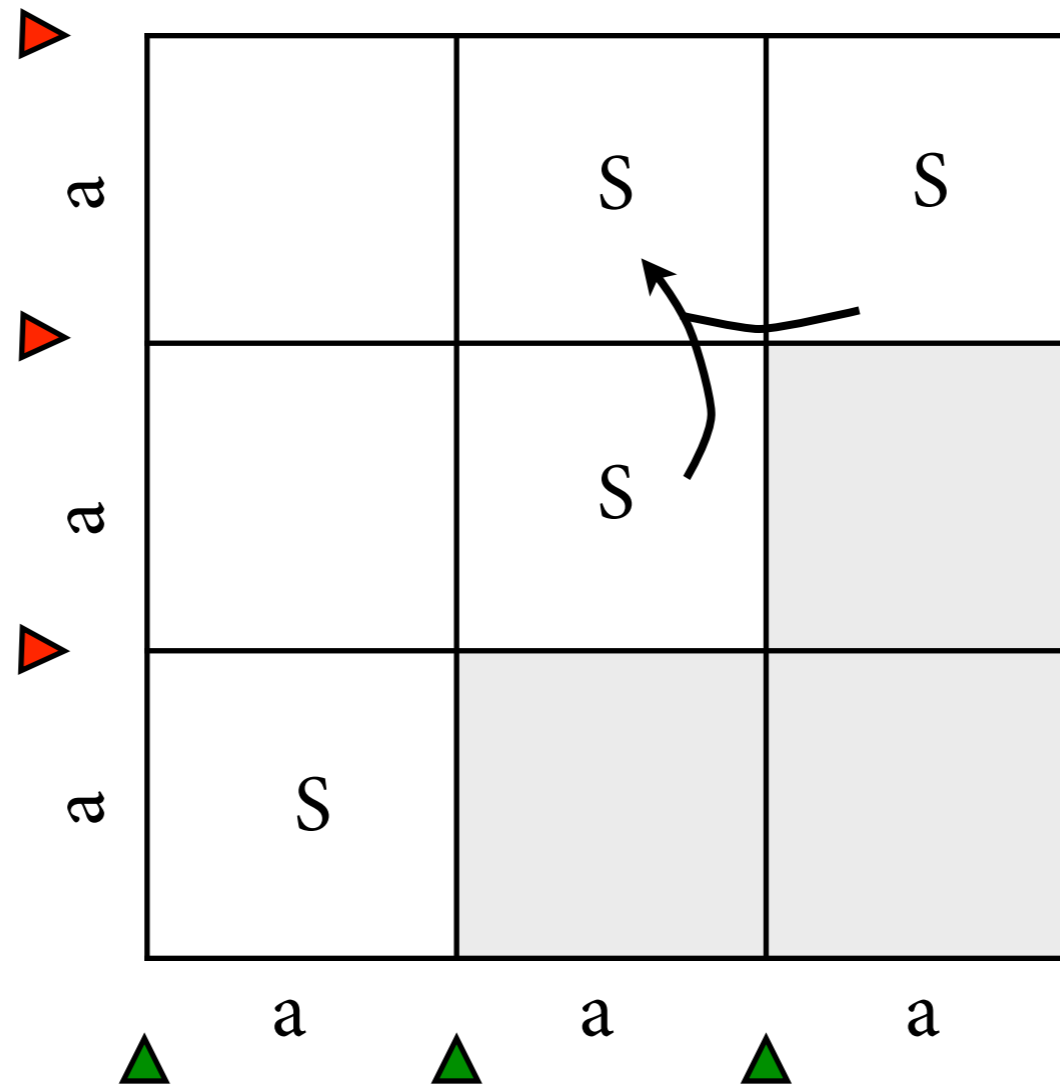
$S \rightarrow a$

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
| a |   |   | S |
| a |   | S |   |
| a | S |   |   |
|   | a | a | a |

# Der CKY-Parser

$S \rightarrow SS$

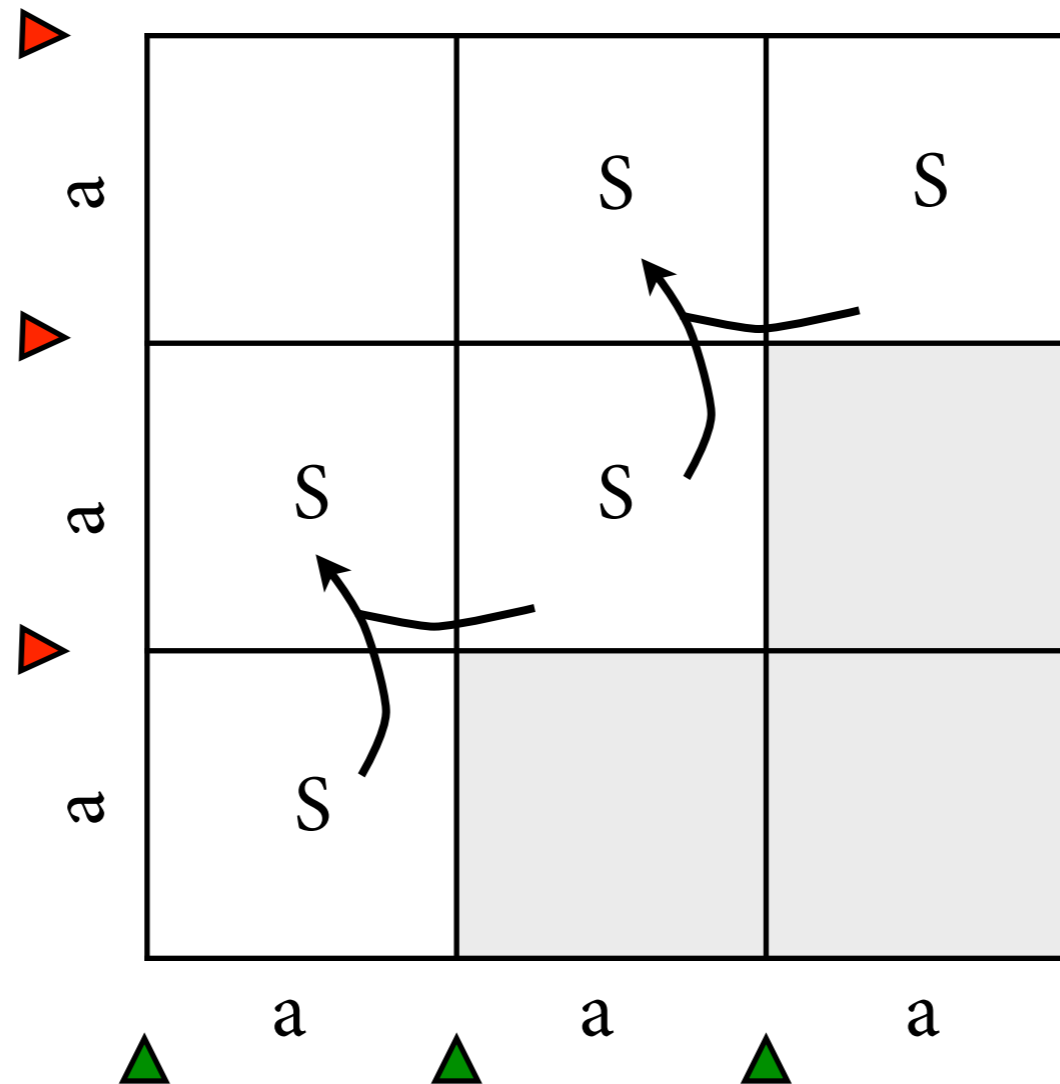
$S \rightarrow a$



# Der CKY-Parser

$S \rightarrow SS$

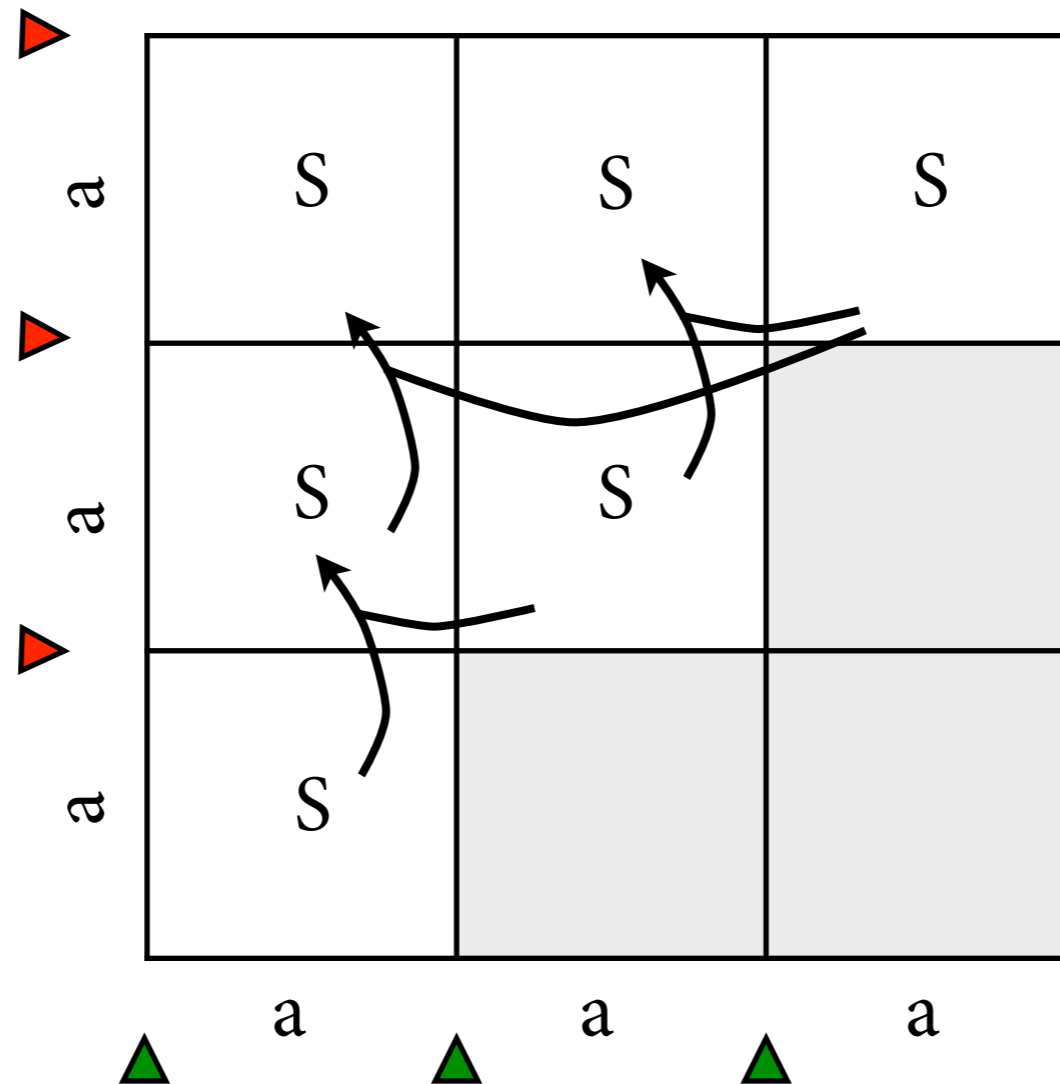
$S \rightarrow a$



# Der CKY-Parser

$S \rightarrow SS$

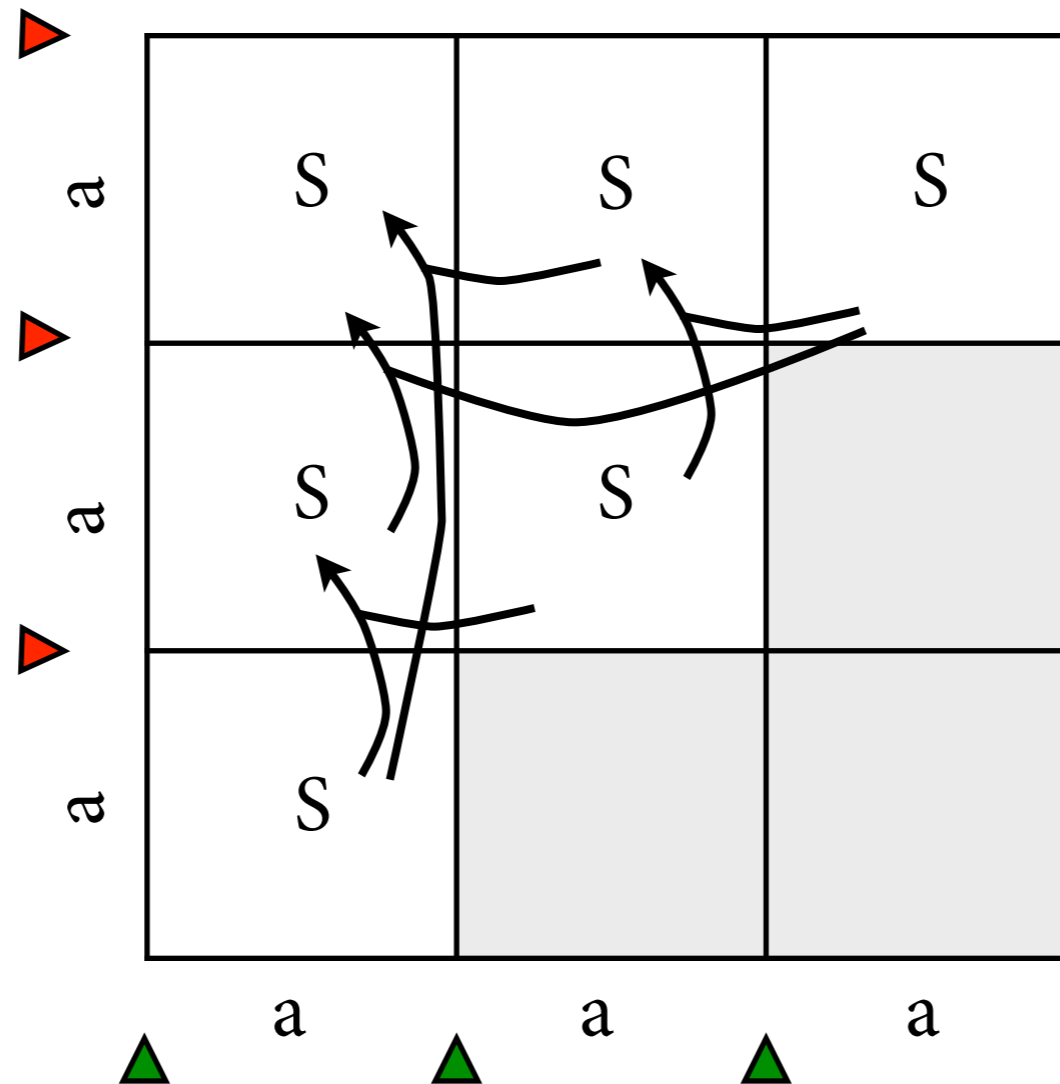
$S \rightarrow a$



# Der CKY-Parser

$S \rightarrow SS$

$S \rightarrow a$



# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Vertical labels on the left:  $\blacktriangledown$  c,  $\blacktriangledown$  b,  $\blacktriangledown$  b,  $\blacktriangledown$  b

Horizontal labels at the bottom:  $\blacktriangle$  b  $\blacktriangle$  b  $\blacktriangle$  b  $\blacktriangle$  c



# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|   |   |  |   |  |   |  |   |
|---|---|--|---|--|---|--|---|
|   |   |  |   |  |   |  |   |
| c |   |  |   |  |   |  |   |
|   |   |  |   |  |   |  |   |
| b |   |  |   |  |   |  |   |
|   |   |  |   |  |   |  |   |
| b |   |  |   |  |   |  |   |
|   |   |  |   |  |   |  |   |
| b | B |  |   |  |   |  |   |
|   | b |  | b |  | b |  | c |

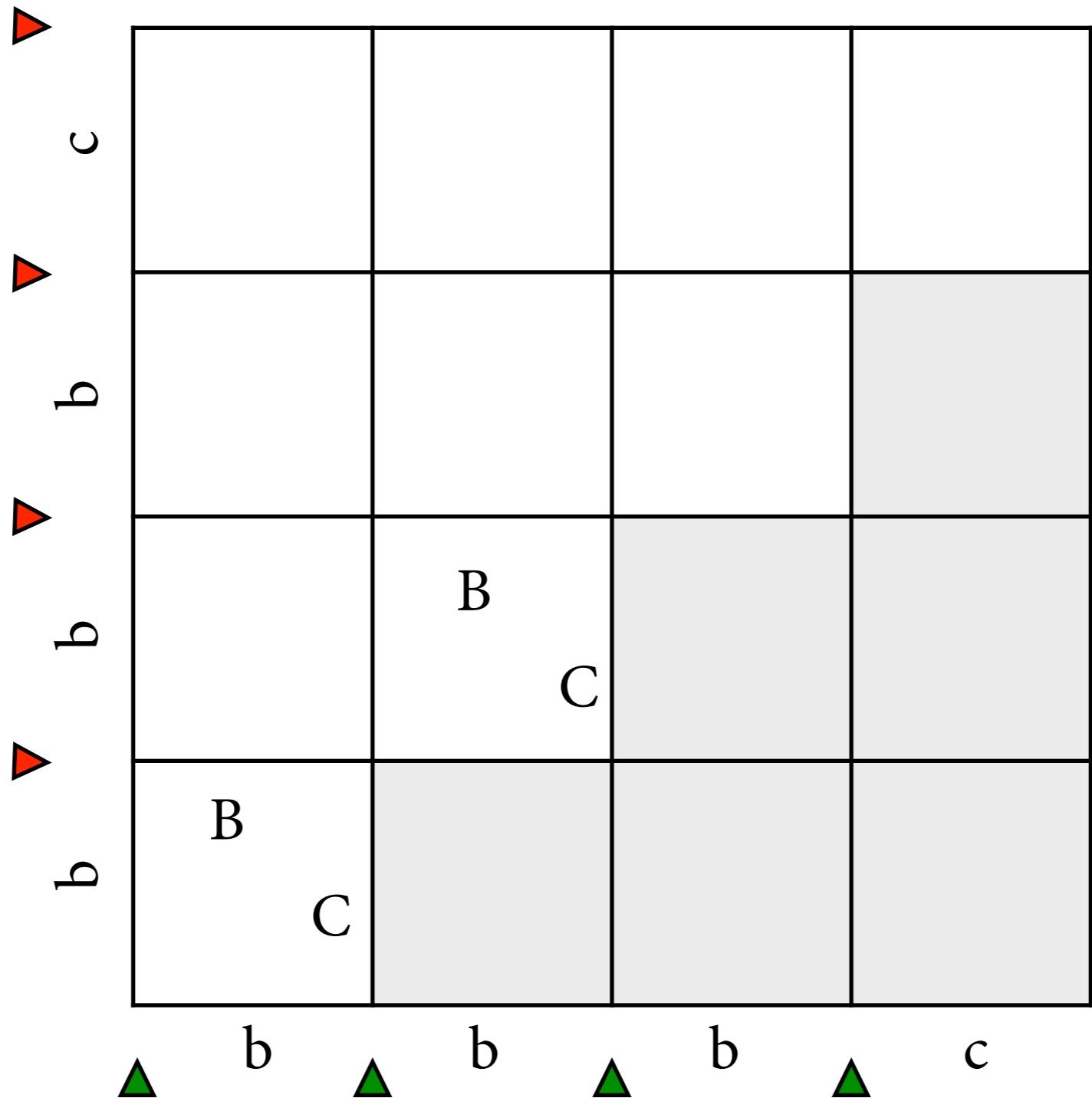
# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| ▶ |   |   |   |   |   |   |   |
| c |   |   |   |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |
| b |   |   |   |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |
| b |   |   |   |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |
| b | B |   |   |   |   |   |   |
|   |   | C |   |   |   |   |   |
| ▲ | b | ▲ | b | ▲ | b | ▲ | c |

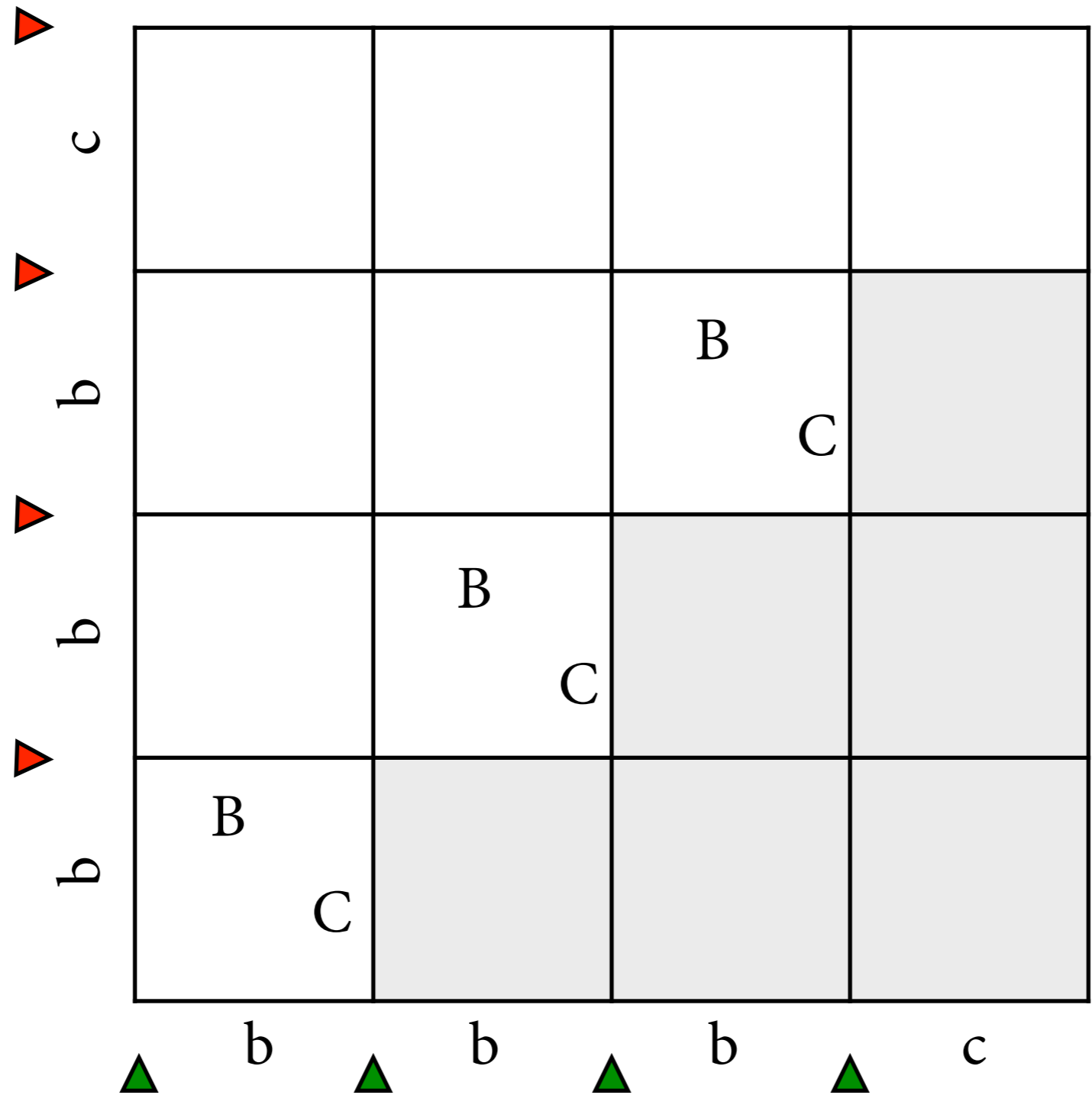
# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |



# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |



# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

A CKY parse table for the string "bbbbc". The table is a 4x4 grid with terminals 'b' and 'c' on the axes. The string "bbbbc" is written below the columns, and "bbbc" is written to the left of the rows. Green triangles point to the first 'b' in each column, and red triangles point to the first 'b' in each row. The table contains non-terminal symbols 'B', 'C', 'S', and 'T' in various cells. Shaded gray cells represent cells that are not part of any parse tree.

|   |   |   |   |   |
|---|---|---|---|---|
|   |   |   | S |   |
|   |   | B |   | T |
|   | B |   |   |   |
| B |   |   |   |   |
| b | b | b | b | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|             |   |   |   |   |
|-------------|---|---|---|---|
|             |   |   | S |   |
|             |   |   |   | T |
|             |   | B |   |   |
|             |   |   | C |   |
| $\emptyset$ | B |   |   |   |
|             |   | C |   |   |
| B           |   |   |   |   |
|             | C |   |   |   |
|             | b | b | b | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|  |   |             |             |   |   |  |   |
|--|---|-------------|-------------|---|---|--|---|
|  |   |             |             | S |   |  | T |
|  |   |             | $\emptyset$ | B |   |  |   |
|  |   | $\emptyset$ | B           |   |   |  |   |
|  |   |             |             | C |   |  |   |
|  | B |             |             |   |   |  |   |
|  | b | C           |             |   |   |  |   |
|  | b |             | b           |   | b |  | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| ▶ |   |   | S | S |   |   |   |
| c |   |   |   |   |   |   | T |
| ▶ |   |   | ∅ | B | C |   |   |
| b |   | ∅ | B |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |
| b | B |   |   |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |
| b |   | C |   |   |   |   |   |
|   | ▲ | b | ▲ | b | ▲ | b | ▲ |
|   |   |   |   |   |   |   | c |



# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| ▶ |   |   | S | S |   |
| ▶ | c |   | T | T |   |
| ▶ | b | ∅ | B |   |   |
| ▶ | b | ∅ | C |   |   |
| ▶ | b | B |   |   |   |
| ▶ | b | C |   |   |   |
| ▲ | b | ▲ | b | ▲ | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

A CKY parse table for the string "bbbc". The table is a 4x4 grid with terminals 'b' and 'c' on the axes. The top-left cell is empty. The top row contains 'S' in the second and third columns, and 'S' in the fourth column. The second row contains '∅' in the first and second columns, 'B' in the third column, and a shaded cell in the fourth column. The third row contains '∅' in the first column, 'B' in the second column, and a shaded cell in the third and fourth columns. The bottom row contains 'B' in the first column, 'C' in the second column, and shaded cells in the third and fourth columns. Red triangles point to the top of each row, and green triangles point to the bottom of each column.

|   |   |   |   |
|---|---|---|---|
|   |   | S | S |
| ∅ | ∅ | B |   |
| ∅ | B |   |   |
| B | C |   |   |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| ▶ |   | S | S | S |   |   |   |
| c |   |   | T | T |   |   |   |
| ▶ | ∅ | ∅ | B |   |   |   |   |
| b |   |   | C |   |   |   |   |
| ▶ | ∅ | B |   |   |   |   |   |
| b |   | C |   |   |   |   |   |
| ▶ | B |   |   |   |   |   |   |
| b | C |   |   |   |   |   |   |
| ▲ | b | ▲ | b | ▲ | b | ▲ | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

A CKY parse table for the string "bbbc". The table is a 4x4 grid with terminals 'b' and 'c' on the top and left edges, and non-terminals 'S', 'T', 'B', and 'C' in the cells. Red triangles point to the top edge, and green triangles point to the bottom edge. Shaded cells indicate that no parse trees can be derived for those substrings.

|   |   |   |   |   |
|---|---|---|---|---|
|   |   | S | S | S |
| c |   | T | T | T |
| b | ∅ | ∅ | B |   |
| b | ∅ | B | C |   |
| b | B |   |   |   |
|   | C |   |   |   |
|   | b | b | b | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

A CKY parse table for the string "bbbc". The table is a 4x4 grid with terminals 'b', 'b', 'b', 'c' on the bottom axis and 'b', 'b', 'c' on the left axis. Red triangles point to the left axis labels, and green triangles point to the bottom axis labels. The table contains non-terminals 'S', 'T', 'B', and 'C', empty sets (∅), and shaded cells representing non-derivable substrings.

|  |   |   |   |   |   |  |   |
|--|---|---|---|---|---|--|---|
|  |   | S | S | S | S |  |   |
|  | c |   | T | T | T |  |   |
|  |   | ∅ | ∅ | B |   |  |   |
|  | b |   |   | C |   |  |   |
|  |   | ∅ | B |   |   |  |   |
|  | b |   | C |   |   |  |   |
|  |   | B |   |   |   |  |   |
|  | b | C |   |   |   |  |   |
|  |   |   |   |   |   |  |   |
|  | b |   | b |   | b |  | c |

# Der CKY-Parser

|                    |                   |                   |
|--------------------|-------------------|-------------------|
| $S \rightarrow BS$ | $B \rightarrow b$ | $S \rightarrow c$ |
| $T \rightarrow CT$ | $C \rightarrow b$ | $T \rightarrow c$ |

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| ▶ |   |   |   |   |   |   |   |   |
| c | S | T | S | T | S | T | S | T |
| ▶ |   | ∅ | ∅ | B |   |   |   |   |
| b |   | ∅ | B | C |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |   |
| b |   | ∅ | B | C |   |   |   |   |
| ▶ |   |   |   |   |   |   |   |   |
| b | B | C |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
|   | ▲ | b | ▲ | b | ▲ | b | ▲ | c |

# CKY-Parser: Algorithmus

Datenstruktur:  $Ch(i,k)$  enthält Menge aller Nichtterminale  $A$  mit  $A \Rightarrow^* w_i \dots w_{k-1}$  (anfangs überall leer).

für alle  $i$  von 1 bis  $n$ :

  für alle Produktionen  $A \rightarrow w_i$ :

    füge  $A$  zu  $Ch(i, i+1)$  hinzu

für alle  $b$  von 2 bis  $n$ :

  für alle  $i$  von 1 bis  $n-b+1$ :

    für alle  $k$  von 1 bis  $b-1$ :

      für alle  $B \in Ch(i, i+k)$  und  $C \in Ch(i+k, i+b)$ :

        für alle Produktionen  $A \rightarrow B C$ :

          füge  $A$  zu  $Ch(i, i+b)$  hinzu

# CKY-Parser

- Erkennen: Wort in Sprache gdw am Schluss das Startsymbol in  $Ch(1, n+1)$  steht.
- Parser: Muss sich für jeden Eintrag in der Tabelle merken, wie man ihn aus kleineren Einträgen bauen kann.



# Laufzeit

- Jede Menge  $Ch(i,k)$  kann höchstens so viele Elemente haben, wie es Nichtterminale gibt (= konstant in der Eingabelänge).
- Es bleiben drei Schleifen ( $b, i, k$ ) über die Eingabelänge  $n$ .
- Deshalb terminiert CKY-Algorithmus unbedingt nach  $O(n^3)$  Schritten: polynomieller Erkenner!
  - ▶ mal Faktor für die Größe der Grammatik (hängt aber nicht von der Eingabelänge ab)

# Korrektheit

- Zu zeigen: Wenn Parser  $A$  zu  $\text{Ch}(i,k)$  hinzufügt, dann gilt  $A \Rightarrow^* w_i \dots w_{k-1}$ .
- Vollständige Induktion über  $k-i$ :
  - ▶  $k-i = 1$ : Folgt aus Regeln für Terminalsymbole.
  - ▶ CKY-Parser fügt  $A$  zu  $\text{Ch}(i,k)$  nur dann hinzu, wenn es Regel  $A \rightarrow B C$  und  $i < j < k$  gibt mit  $B \in \text{Ch}(i,j)$  und  $C \in \text{Ch}(j,k)$ .
  - ▶ Wenn Aussage also für  $\text{Ch}(i,j)$  und  $\text{Ch}(j,k)$  gilt, dann ist  $A \Rightarrow B C \Rightarrow^* w_i \dots w_{j-1} C \Rightarrow^* w_i \dots w_{j-1} w_j \dots w_{k-1}$

# Vollständigkeit

- Zu zeigen: Wenn  $A \Rightarrow^* w_i \dots w_{k-1}$ , dann legt Parser irgendwann  $A$  nach  $Ch(i,k)$ .
- Knackpunkt im Beweis: Zum Zeitpunkt der Berechnung von  $Ch(i,k)$  stehen alle Einträge, die kürzer als  $k-i$  sind, schon in der Chart.
- Das wird durch die Reihenfolge der Schleifen im CKY-Algorithmus garantiert.
  - ▶ äußerste Schleife geht über Breite des Teilstrings
  - ▶ innere Schleifen über Anfangsposition, Breite des linken Teils

# Implementierung

- Idee der Datenstruktur:
  - ▶  $\text{chart}[i][k] = \text{Ch}(i+1, k+1)$
  - ▶ chart ist eine Liste von Zeilen der Charts
  - ▶ Jede Zeile ist eine Liste von Mengen von NT-Symbolen.

- Initialisierung:

```
chart = []
```

```
for i in range(n+1):  
    row = []  
    for j in range(n+1):  
        row.append(set())  
    chart.append(row)
```

# Implementierung

```
for i in range(n):                                # Terminalregeln
    for prod in grammar productions(rhs=words[i]):
        chart[i][i+1].add(prod.lhs())
```

```
for width in range(2, n+1):                       # Binaere Regeln
    for i in range(0, n-width+1):
        for j in range(1, width):
            nts1 = chart[i][i+j]
            nts2 = chart[i+j][i+width]
            for nt1 in nts1:
                productions = grammar productions(rhs=nt1)
                for production in productions:
                    if production.rhs()[1] in nts2:
                        chart[i][i+width].add(production.lhs())
```

# Implementierung: Parser

- Für einen *Parser* merkt man sich zu jedem Nichtterminal in einer Chartzelle, auf welche (mehreren) Weisen man es bauen kann.
- `chart[i][k]` ist jetzt Dictionary (statt Menge).
  - ▶ `keys(chart[i][k])`: Nichtterminale  $A$ , die den Teilstring  $w_i \dots w_{k-1}$  abdecken können
  - ▶ Eintrag `chart[i][k][“A”]`: Liste von *Backpointern*, d.h. Tripeln  $(B, C, j)$ , die angeben, dass  $A$  aus  $B$  von  $i$  bis  $j$  plus  $C$  von  $j$  bis  $k$  gebaut werden kann.
- Daraus mit rekursiver Funktion Bäume ausrechnen.

# Zusammenfassung

- Rekursionsbasierte Parser (RD, SR) können exponentielle Laufzeit brauchen.
  - ▶ in der Praxis viel zu langsam
- CKY ist polynomieller Erkennenner.
  - ▶ verwendet *Chart*, um Zwischenergebnisse zu tabellieren.
- Laufzeit reduziert sich auf  $O(n^3)$ .