Natural Language Processing

Lecture 6: Parsing with Context Free Grammars I.
CKY algorithm

09/25/2018

COMS W4705
Daniel Bauer

Formal Grammar and Parsing

- Formal Grammars are used in linguistics, NLP, programming languages.
- We want to build a compact model that describes a complete language.
- Need efficient algorithms to determine if a sentence is in the language or not **(recognition problem)**.
- We also want to recover the structure imposed by the grammar (**parsing problem**).

Syntactic Parsing

- Formalisms like CFGs and Finite State Automata define the (possibly infinite) set of legal strings of a language.

- **Parsing algorithms** determine if an input string is part of this language or not. For CFGs, they assign each string one or more syntactic analyses.

Two Approaches to Parsing

- Bottom-up: Start at the words (terminal symbols) and see which subtrees you can build. Then combine these subtrees into larger trees. (Driven by the input sentence.)
  CKY algorithm - requires Grammars in Chomsky Normal Form.

- Top-down: Start at the start symbol (S), try to apply production rules that are compatible with the input.
  (Driven by the grammar - next week)
  Earley algorithm

- Both approaches can be seen as a kind of search problem (next week).

Chomsky Normal Form

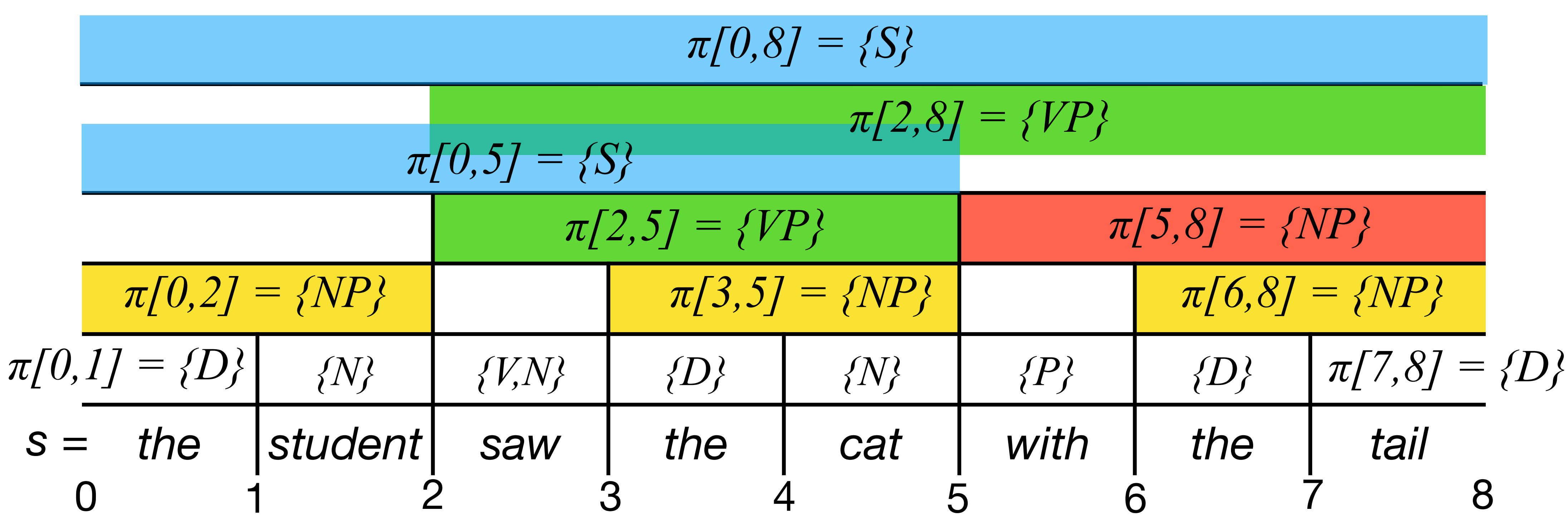
- A CFG $G=(N, \Sigma, R, S)$ is in Chomsky Normal Form (CNF) if the rules take one of the following forms:
  - $A \rightarrow B\ C$, where $A \in N, B \in N, C \in N$.
  - $A \rightarrow b$, where $A \in N, b \in \Sigma$.

S	→	NP VP	V	→	saw
VP	→	V NP	P	→	with
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	tail
NP	→	NP PP	N	→	student

Any CFG can be converted to an equivalent grammar in CNF that expresses the same language.

Cocke-Kasami-Younger (CKY) Algorithm - Motivation

- A nonterminal A *covers* a sub-span $[i,j]$ of the input string s if the rules in the grammar can derive $s[i,j]$ from A.
  Let $\pi[i,j]$ be the set of nonterminals that cover $[i,j]$.
- The string is recognized by the grammar if $S \in \pi[i,j]$.
- Approach: Compute $\pi[i,j]$ for all sub-spans bottom-up, using dynamic-programming.



CKY Data Structure

- Use a 2-dimensional “parse table” to represent $\pi[i,j]$.

S	→ NP VP		NP	→ she
VP	→ V NP		NP	→ glasses
VP	→ VP PP		D	→ the
PP	→ P NP		N	→ cat
NP	→ D N		N	→ glasses
NP	→ NP PP		V	→ saw
			P	→ with

0 she 1 saw 2 the 3 cat 4 with 5 glasses 6

0		0,1	0,2	0,3	0,4	0,5	0,6
1			1,2	1,3	1,4	1,5	1,6
2				2,3	2,4	2,5	2,6
3					3,4	3,5	3,6
4						4,5	4,6
5							5,6
6							

CKY Initialization

- For i=0…length(s-1):

  $$\pi[i, i+1] = \{A \mid A \rightarrow s[i:i+1] \in R\}$$

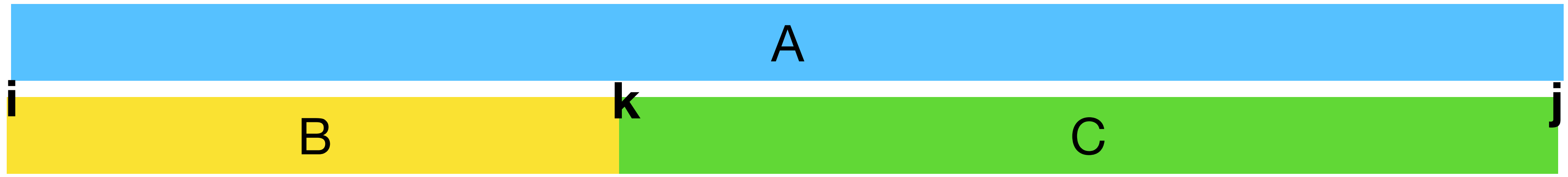
S → NP VP	**NP → she**
VP → V NP	**NP → glasses**
VP → VP PP	**D → the**
PP → P NP	**N → cat**
NP → D N	**N → glasses**
NP → NP PP	**V → saw**
	P → with

$_0$ she $_1$ saw $_2$ the $_3$ cat $_4$ with $_5$ glasses

0		NP	0,2	0,3	0,4	0,5	0,6
1			V	1,3	1,4	1,5	1,6
2				D	2,4	2,5	2,6
3					N	3,5	3,6
4						P	4,6
5							NP,N
6							

CKY - finding the split

- CKY requires grammar to be in CNF.
- Assume subspan [i,j] is covered by nonterminal A.
  - Then this nonterminal was recognized by some production of the form $A \rightarrow B\ C$, where $A \in N, B \in N, C \in N$ (grammar is in CNF).
  - Span $[i,j]$ can be split into two parts:
    $[i,k]$, which is covered by B, and
    $[k,j]$ which is covered by C.



CKY - Recursive Definition

- To compute $\pi[i, j]$, try all possible split points k, such that $i < k < j$.
  - For each k, check if the nonterminals in $\pi[i,k]$ and $\pi[k,j]$ match any of the rules in the grammar.
- Recursive definition for $\pi[i, j]$:

$$\pi[i, j] = \bigcup_{k=i+1\ldots j-1} \{A | A \rightarrow B\ C \in R \text{ and } B \in \pi[i, k] \text{ and } C \in \pi[k, j]\}$$

CKY Full Algorithm

- **Input:** Grammar $G=(N, \Sigma, R, S)$, input string s of length n.

- for i=0…n-1: initialization

  $\pi[i, i+1] = \{A \mid A \rightarrow s[i]\}$

- for $length=2...n$: main loop

  for $i=0...(n\text{-}length)$:

  $j = i+length$

  for $k=i+1...j\text{-}1$:

  $$M = \{A | A \rightarrow B\ C \in R \text{ and } B \in \pi[i,k] \text{ and } C \in \pi[k,j]\}$$

  $$\pi[i,j] = \pi[i,j] \cup M$$

- if $S \in \pi[0, i+1]$ return True, otherwise False

CKY Algorithm

for *i=0…(n-length):*

j = i+length

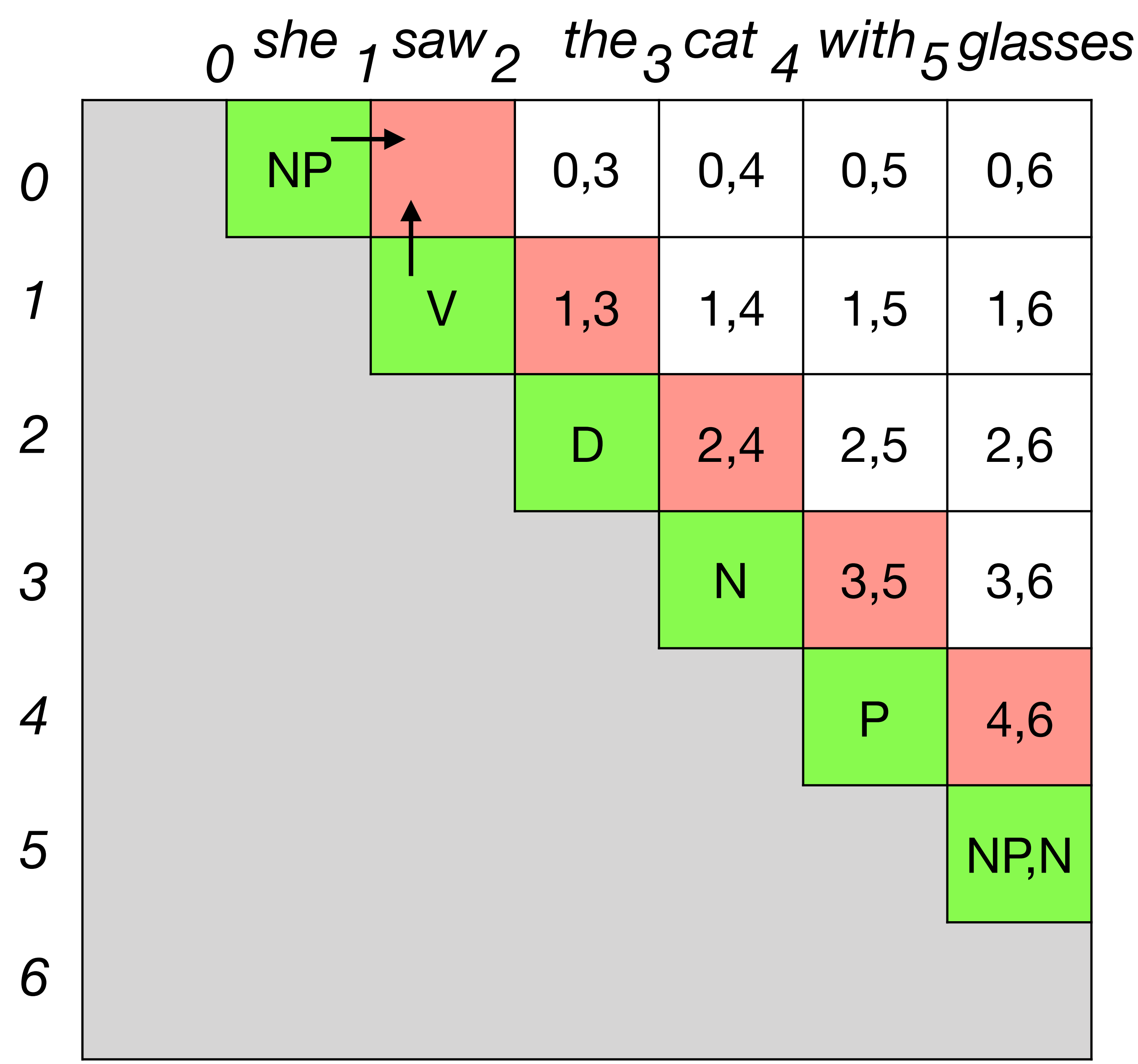
for *k=i+1…j-1:*

….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=2

i=0,k=1,j=2



CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

$\quad j = i+length$

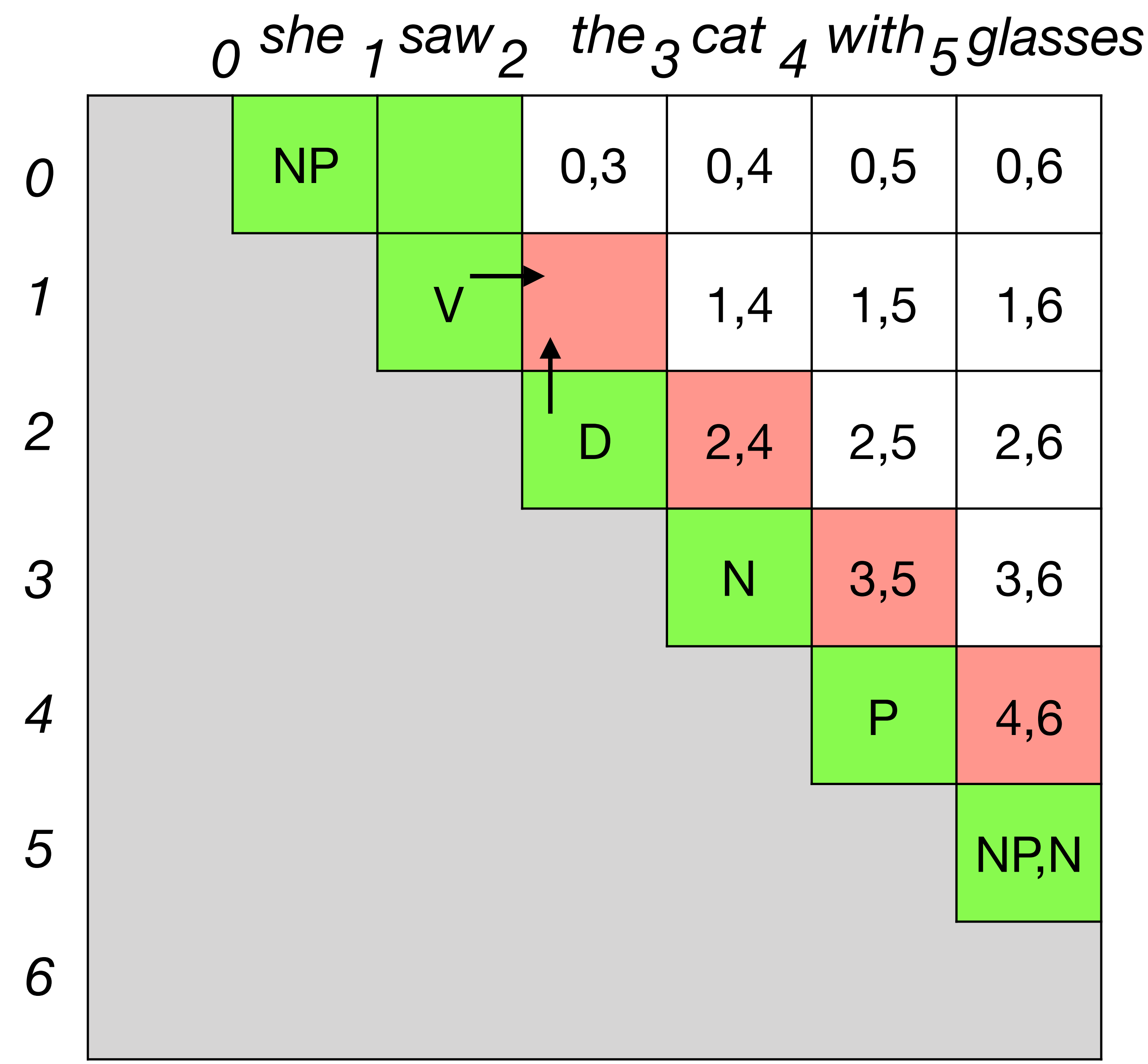
$\quad$ for $k=i+1 \ldots j\text{-}1$:

$\quad\quad \ldots.$

S	→	NP VP	NP	→	she
VP	→	V NP	NP	→	glasses
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	glasses
NP	→	NP PP	V	→	saw
			P	→	with

$length=2$

$i=1, k=2, j=3$



CKY Algorithm

for i=0…(n-length):
  $j = i$+length
  for k=i+1…j-1:
   ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=2

i=2,k=3,j=4

0 she 1 saw 2 the 3 cat 4 with 5 glasses

0	NP		0,3	0,4	0,5	0,6
1		V		1,4	1,5	1,6
2			D	NP	2,5	2,6
3				N	3,5	3,6
4					P	4,6
5						NP,N
6						

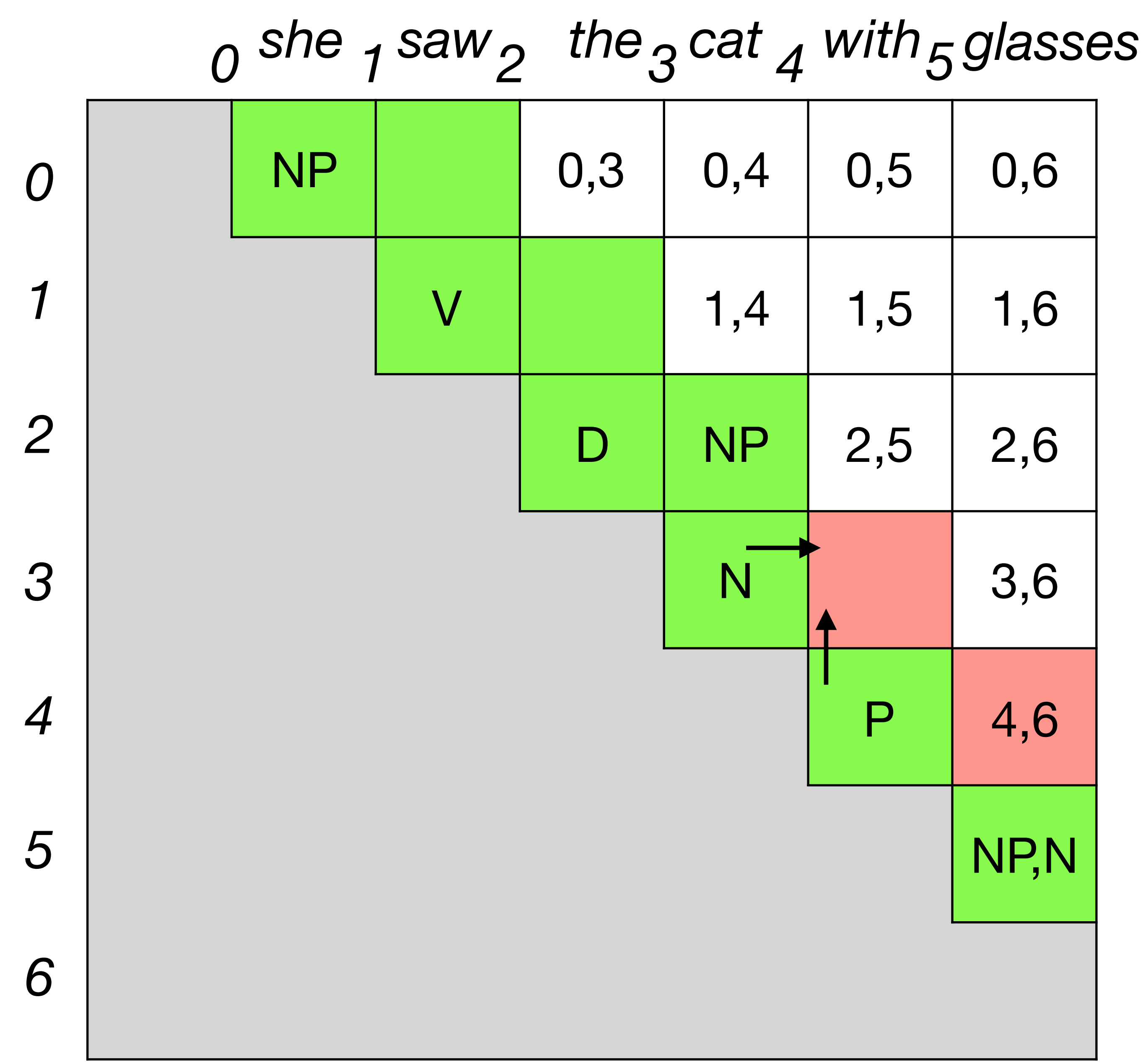
CKY Algorithm

for i=$0 \ldots (n\text{-}length)$:
    $j = i + length$
    for k=$i+1 \ldots j\text{-}1$:
        ....

S	→	NP VP	NP	→	she
VP	→	V NP	NP	→	glasses
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	glasses
NP	→	NP PP	V	→	saw
			P	→	with

$length=2$

$i=3, k=4, j=5$



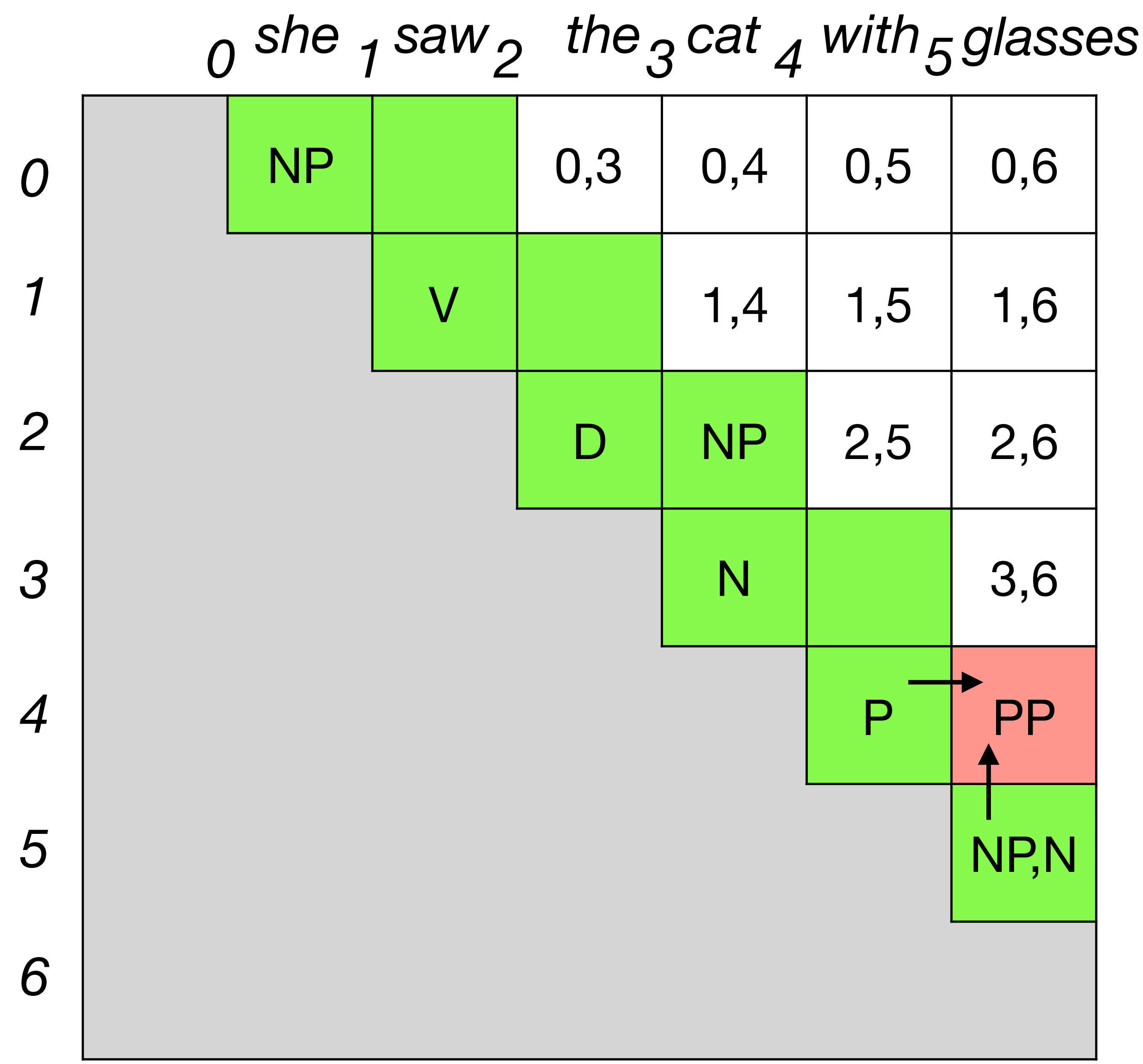
CKY Algorithm

for *i=0…(n-length):*
    j = i+length
    for *k=i+1…j-1:*
        ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=2

i=4,k=5,j=6



CKY Algorithm

for $i=0...(n\text{-}length)$:

  $j = i+length$

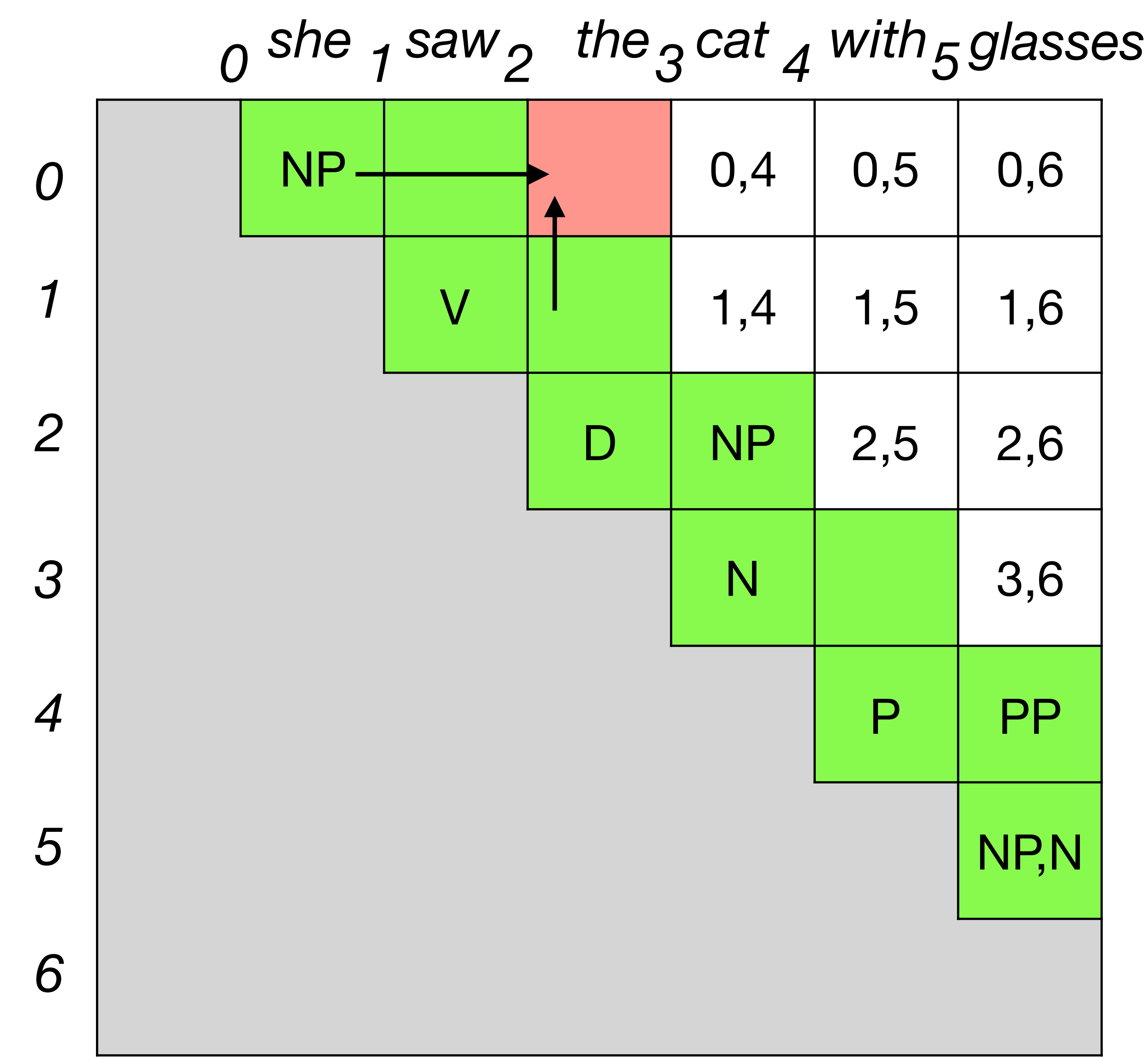
  for $k=i+1...j\text{-}1$:

  ....

S	→	NP VP	NP	→	she
VP	→	V NP	NP	→	glasses
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	glasses
NP	→	NP PP	V	→	saw
			P	→	with

$length=3$

$i=0,k=1,j=3$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

$j = i+length$

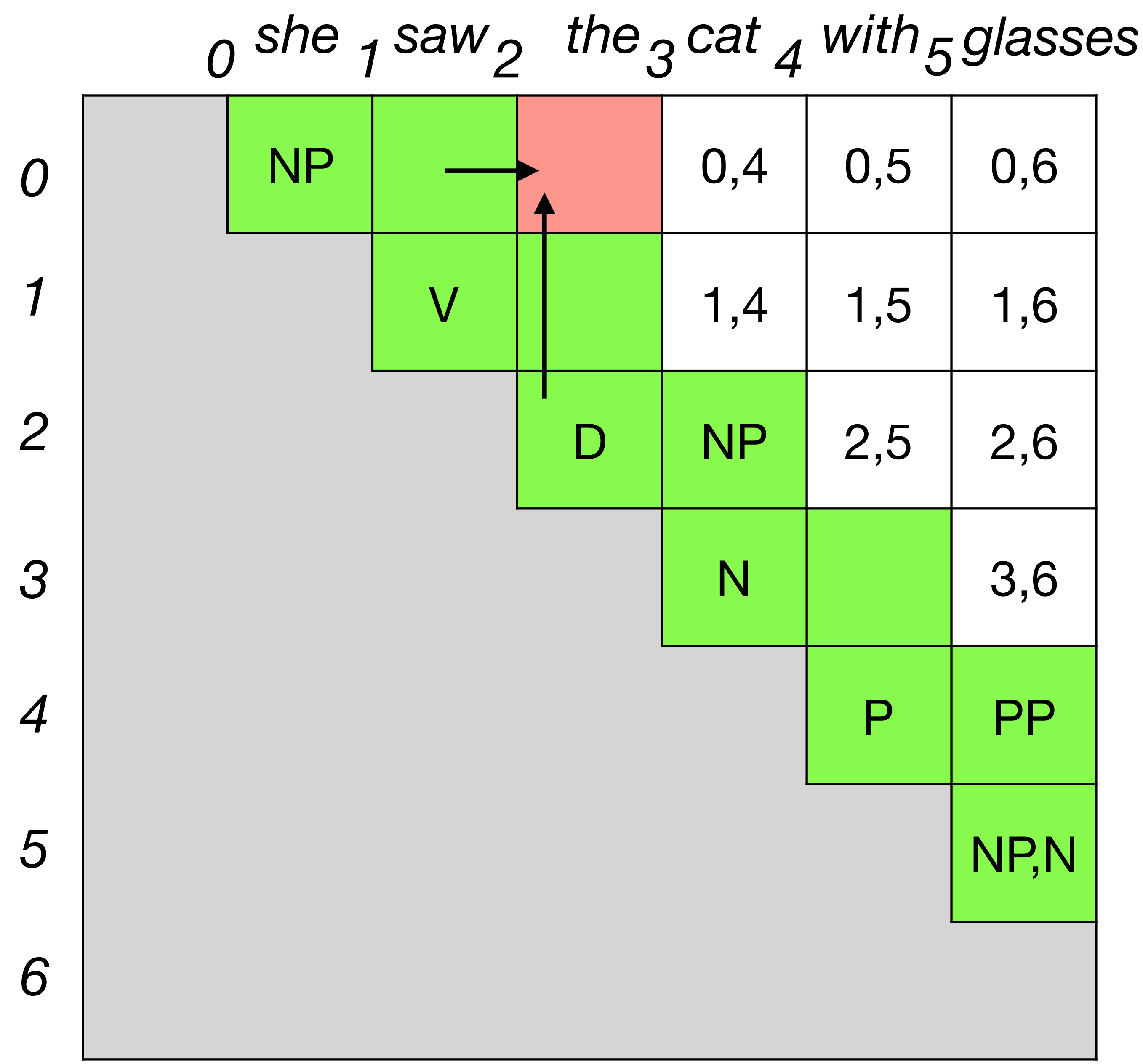
for $k=i+1\ldots j\text{-}1$:

….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=3$

$i=0, k=2, j=3$



CKY Algorithm

for i=0…(n-length):
    j = i+length
    for k=i+1…j-1:
        ….

S → NP VP		NP → she	
VP → V NP		NP → glasses	
VP → VP PP		D → the	
PP → P NP		N → cat	
NP → D N		N → glasses	
NP → NP PP		V → saw	
		P → with	

length=3

i=1,*k*=2,*j*=4

0 she 1 saw 2 the 3 cat 4 with 5 glasses

0		NP			0,4	0,5	0,6
1			V		VP	1,5	1,6
2				D	NP	2,5	2,6
3					N		3,6
4						P	PP
5							NP,N
6							

CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

  $j = i+length$

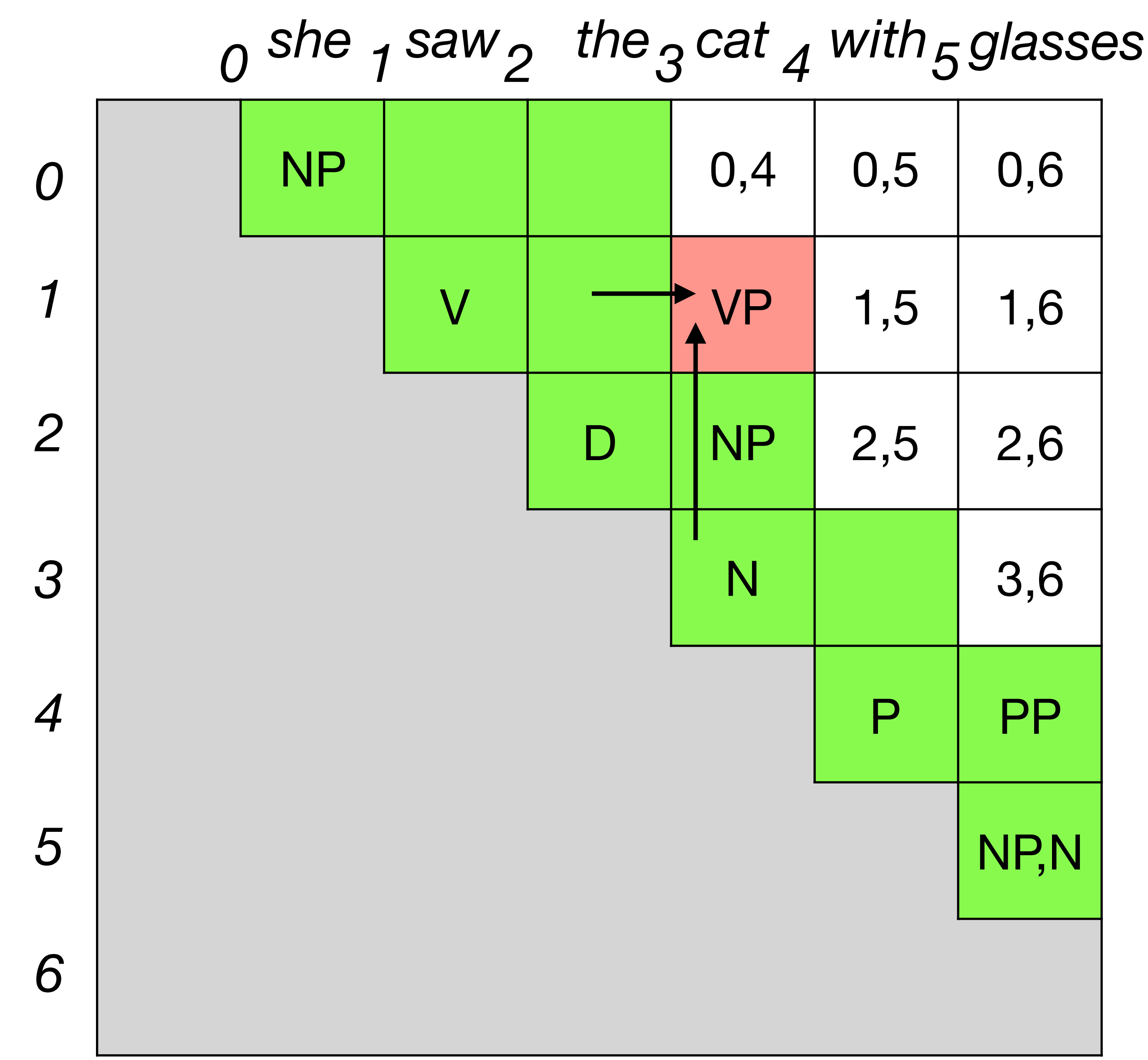
  for $k=i+1 \ldots j\text{-}1$:

  ….

S	→	NP VP	NP	→	she
VP	→	V NP	NP	→	glasses
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	glasses
NP	→	NP PP	V	→	saw
			P	→	with

$length=3$

$i=1, k=3, j=4$



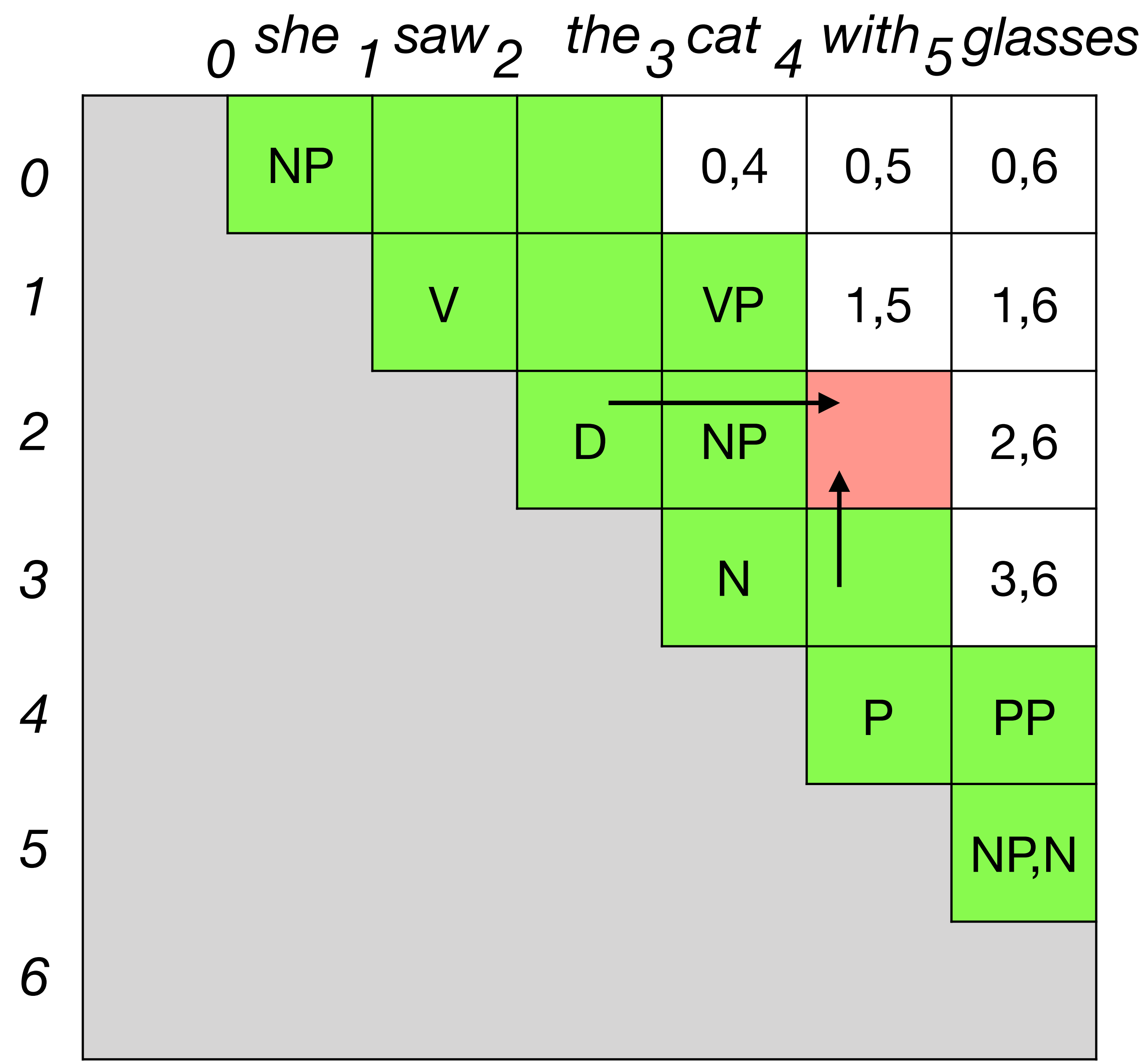
CKY Algorithm

for *i=0…(n-length):*
  j = i+length
  for *k=i+1…j-1:*
  ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=3

i=2,k=3,j=5



CKY Algorithm

for $i=0...(n\text{-}length)$:

   $j = i+length$

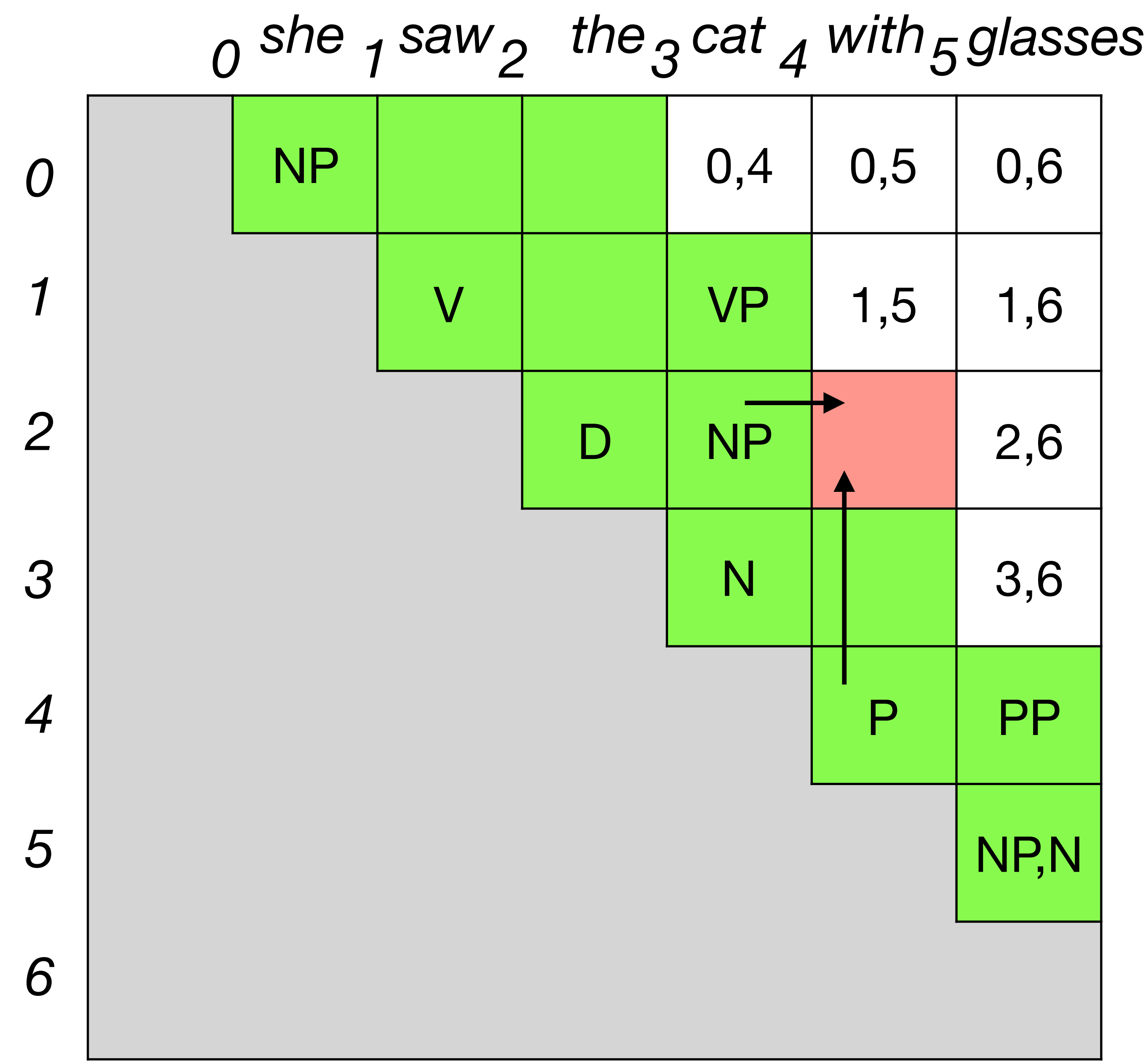
   for $k=i+1...j\text{-}1$:

      ....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=3$

$i=2,k=4,j=5$



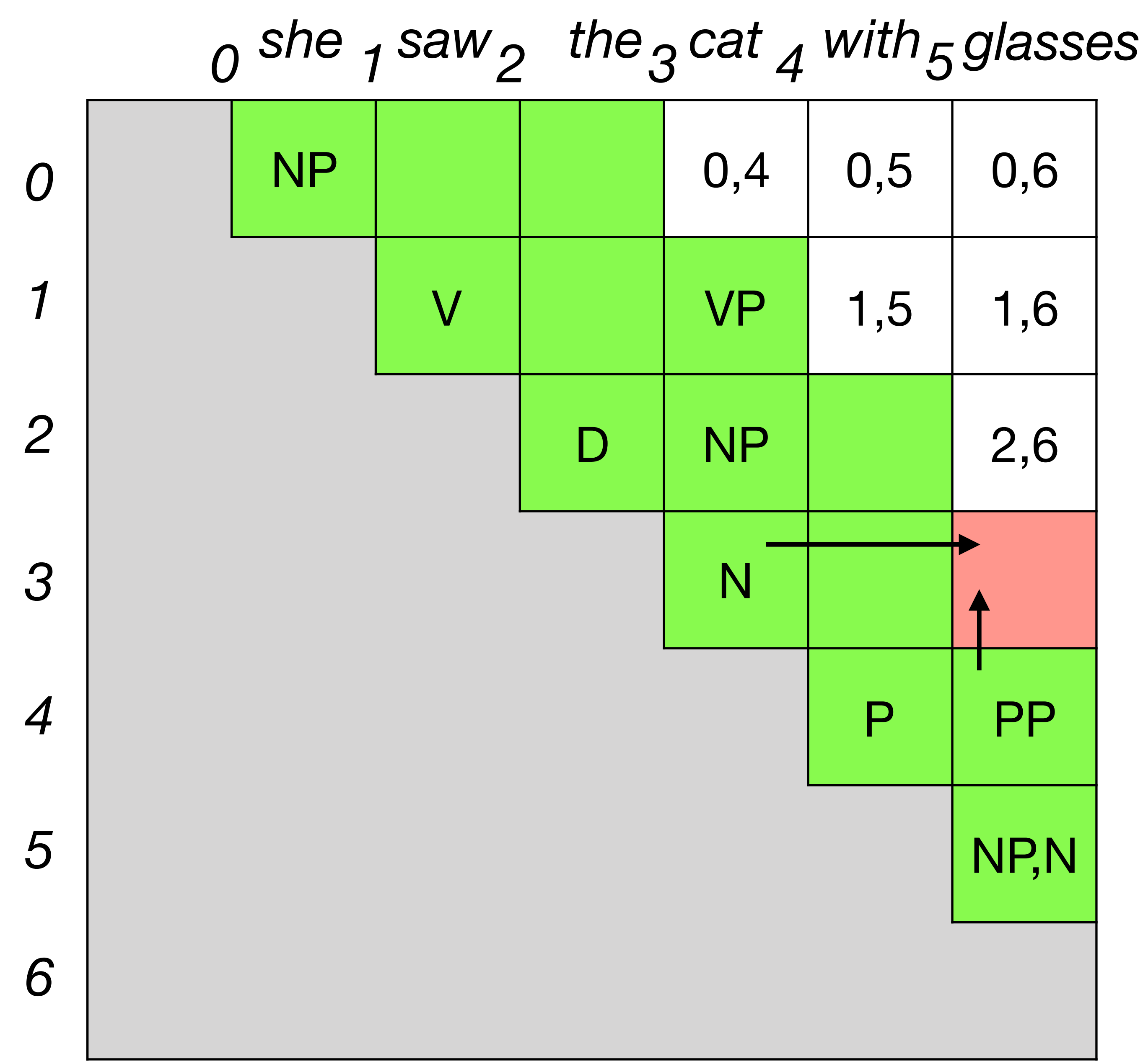
CKY Algorithm

for *i=0…(n-length):*
j = i+length
for *k=i+1…j-1:*
….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=3

i=3,k=4,j=6



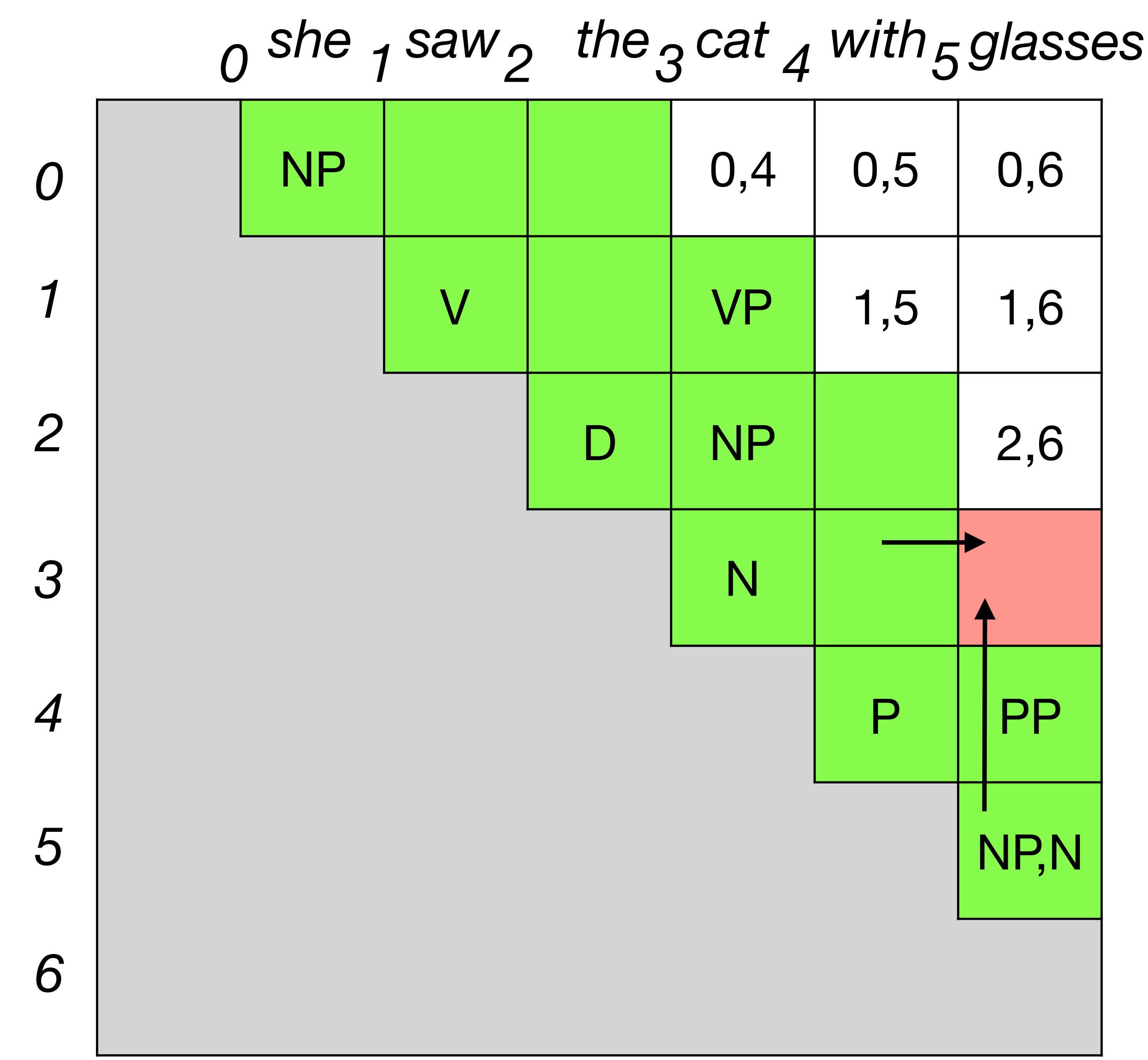
CKY Algorithm

for *i=0…(n-length):*
    j = i+length
    for *k=i+1…j-1:*
        ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=3

i=3,k=5,j=6



CKY Algorithm

for $i=0...(n\text{-}length)$:

$j = i+length$

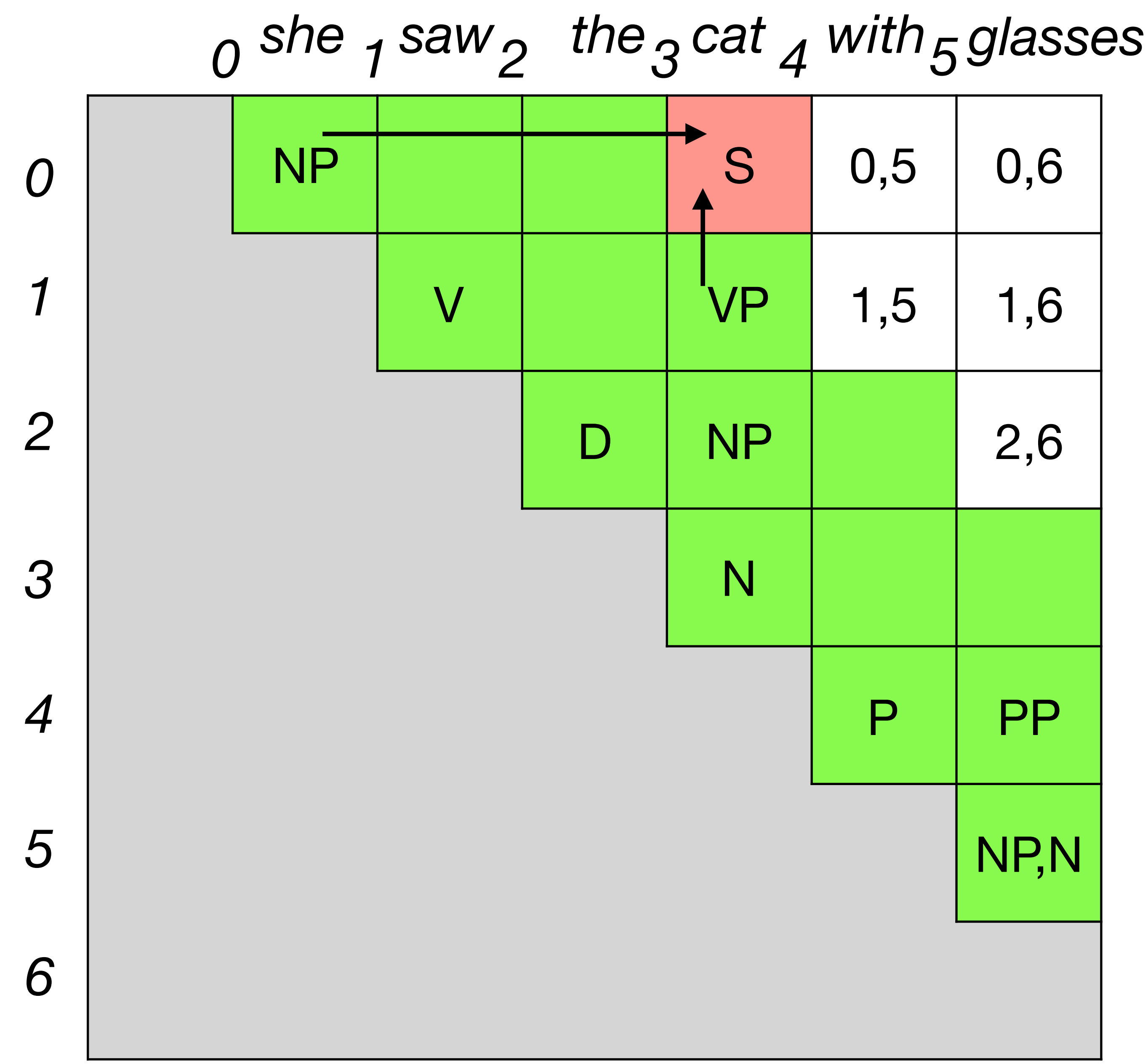
for $k=i+1...j\text{-}1$:

....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=4$

$i=0,k=1,j=4$



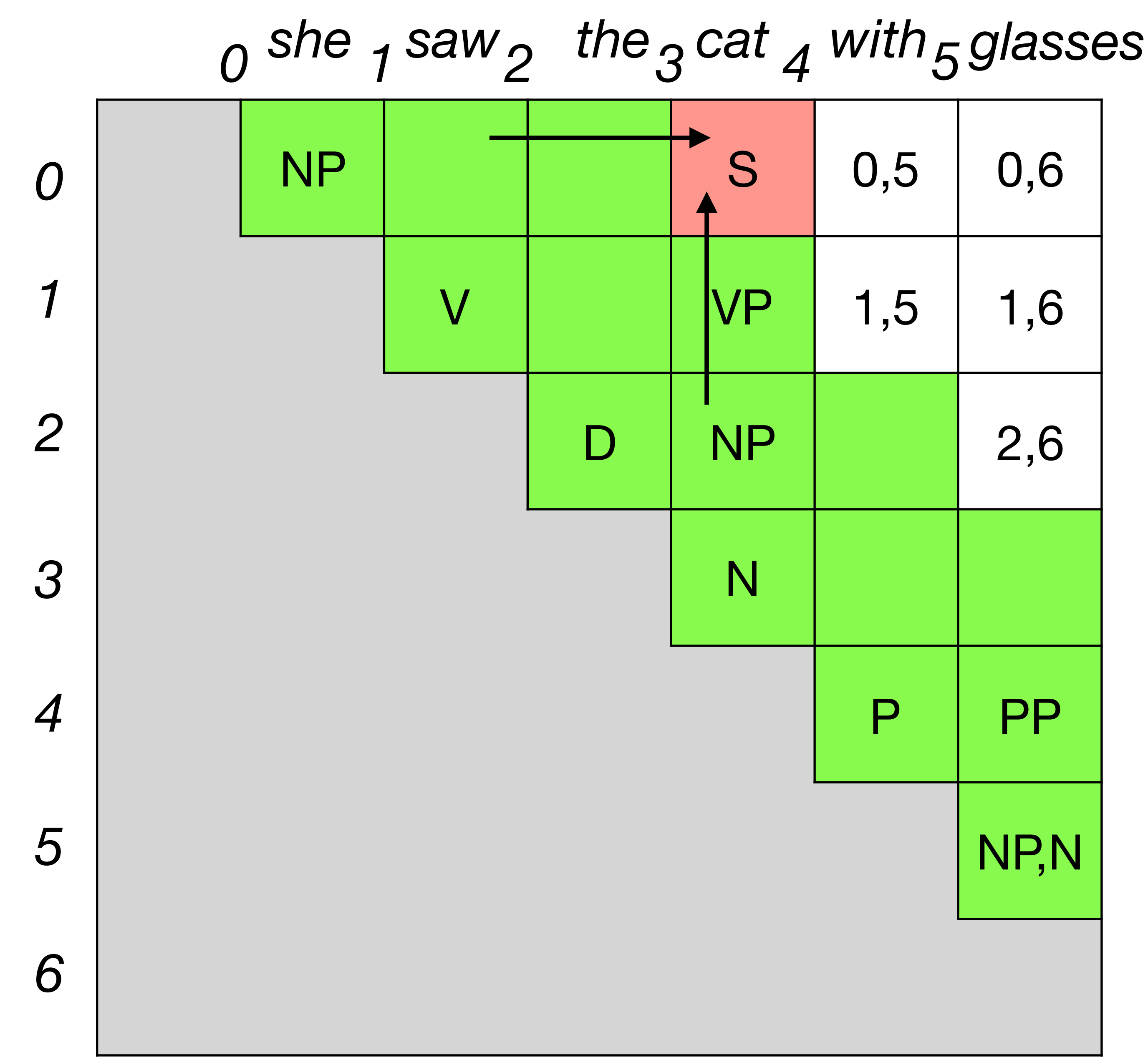
CKY Algorithm

for $i=0...(n\text{-}length)$:
    $j = i+length$
    for $k=i+1...j\text{-}1$:
        ....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=4$

$i=0,k=2,j=4$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

  $j = i+length$

  for $k=i+1\ldots j\text{-}1$:

    ....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=4$

$i=0, k=3, j=4$

0 she 1 saw 2 the 3 cat 4 with 5 glasses

0		NP			S	0,5	0,6
1			V		VP	1,5	1,6
2				D	NP		2,6
3					N		
4						P	PP
5							NP,N
6							

CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

  $j = i+length$

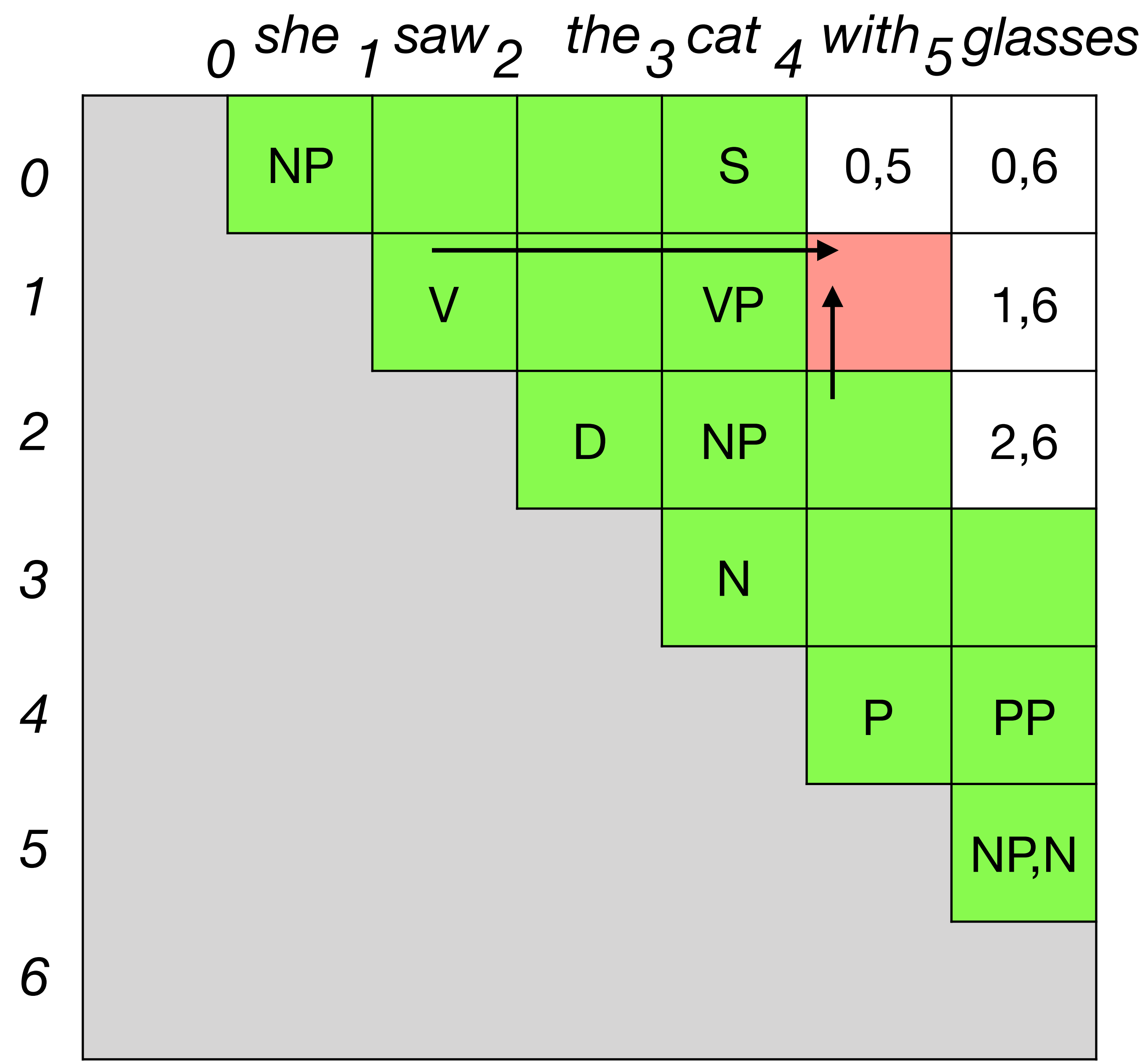
  for $k=i+1\ldots j\text{-}1$:

   ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=4$

$i=1,k=2,j=5$



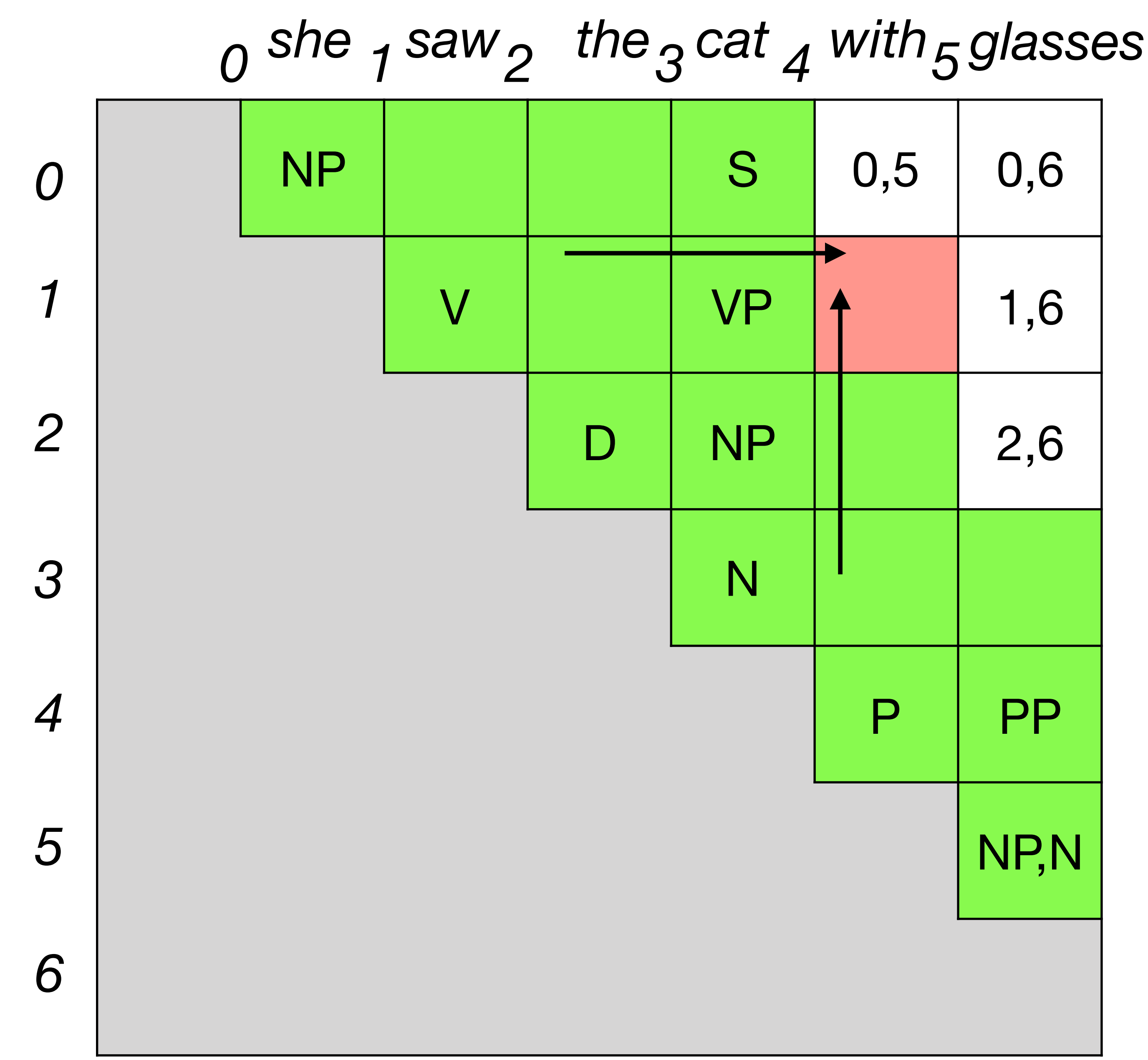
CKY Algorithm

for *i=0…(n-length):*
    j = i+length
    for *k=i+1…j-1:*
        ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=4

i=1,k=3,j=5



CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

$j = i + length$

for $k = i+1 \ldots j-1$:

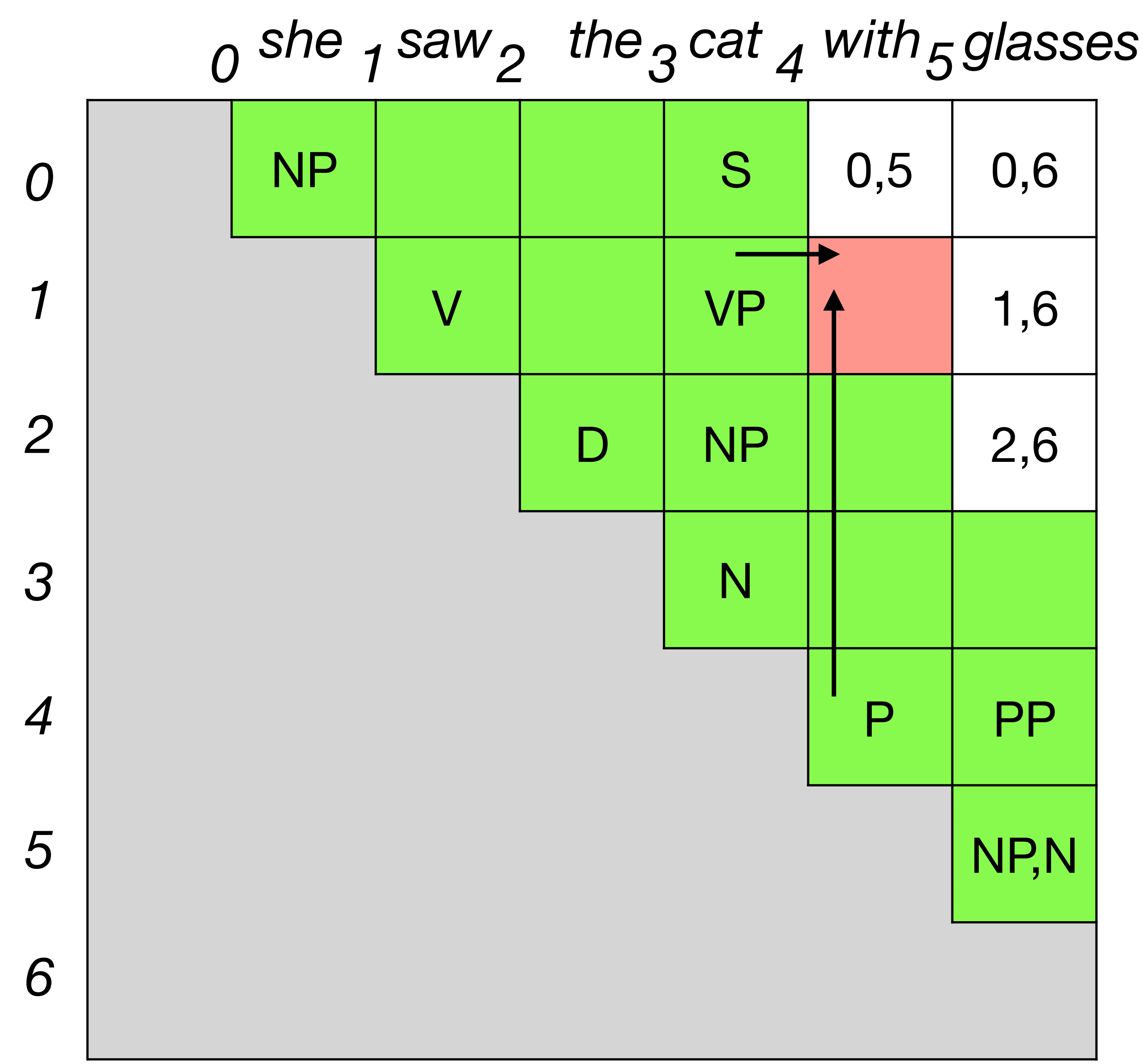
....

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

NP → she
NP → glasses
D → the
N → cat
N → glasses
V → saw
P → with

$length=4$

$i=1, k=4, j=5$



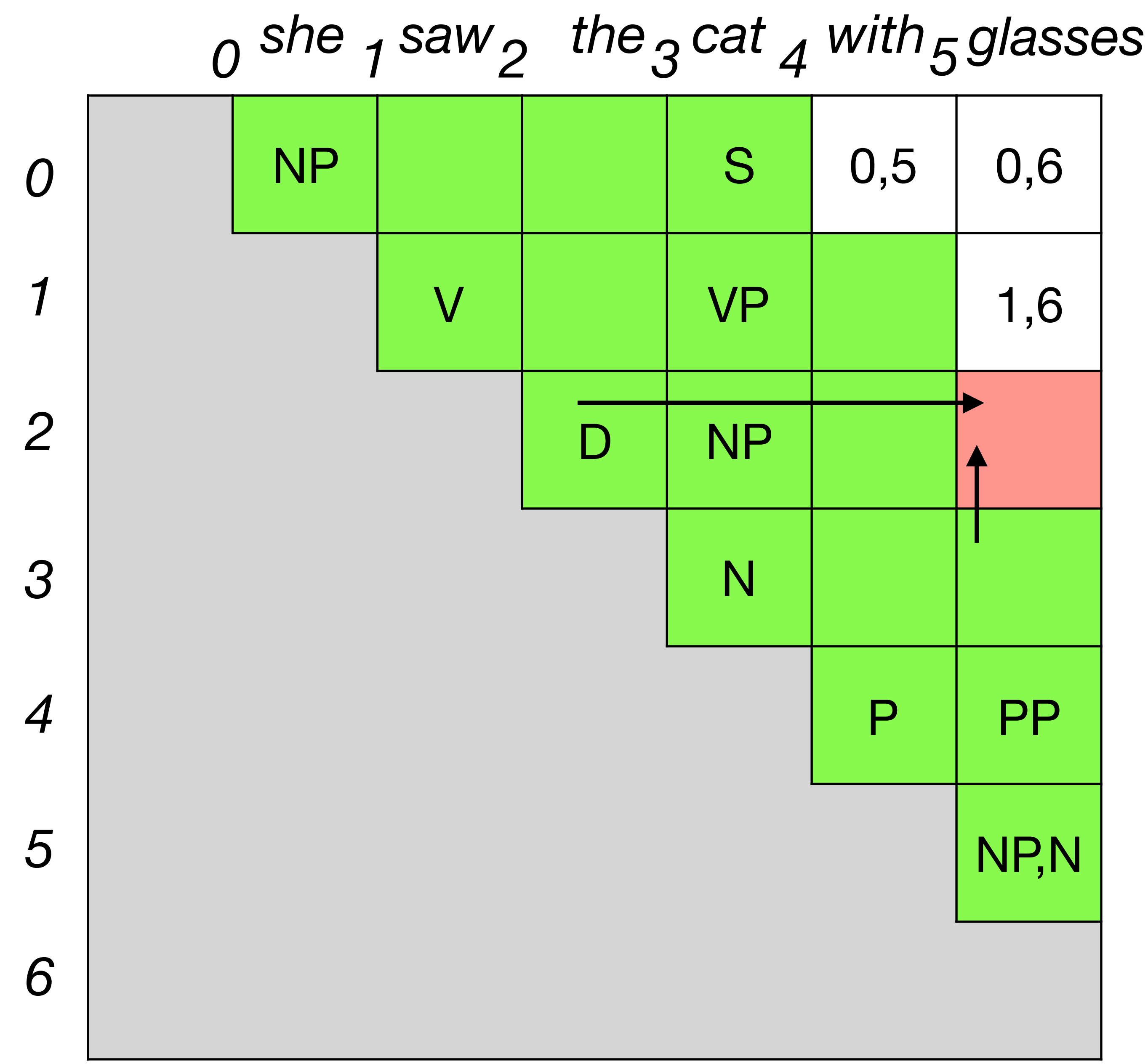
CKY Algorithm

for *i=0…(n-length):*
    j = i+length
    for *k=i+1…j-1:*
        ….

S	→ NP VP	NP	→ she
VP	→ V NP	NP	→ glasses
VP	→ VP PP	D	→ the
PP	→ P NP	N	→ cat
NP	→ D N	N	→ glasses
NP	→ NP PP	V	→ saw
		P	→ with

length=4

i=2,k=3,j=6



CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

  $j = i+length$

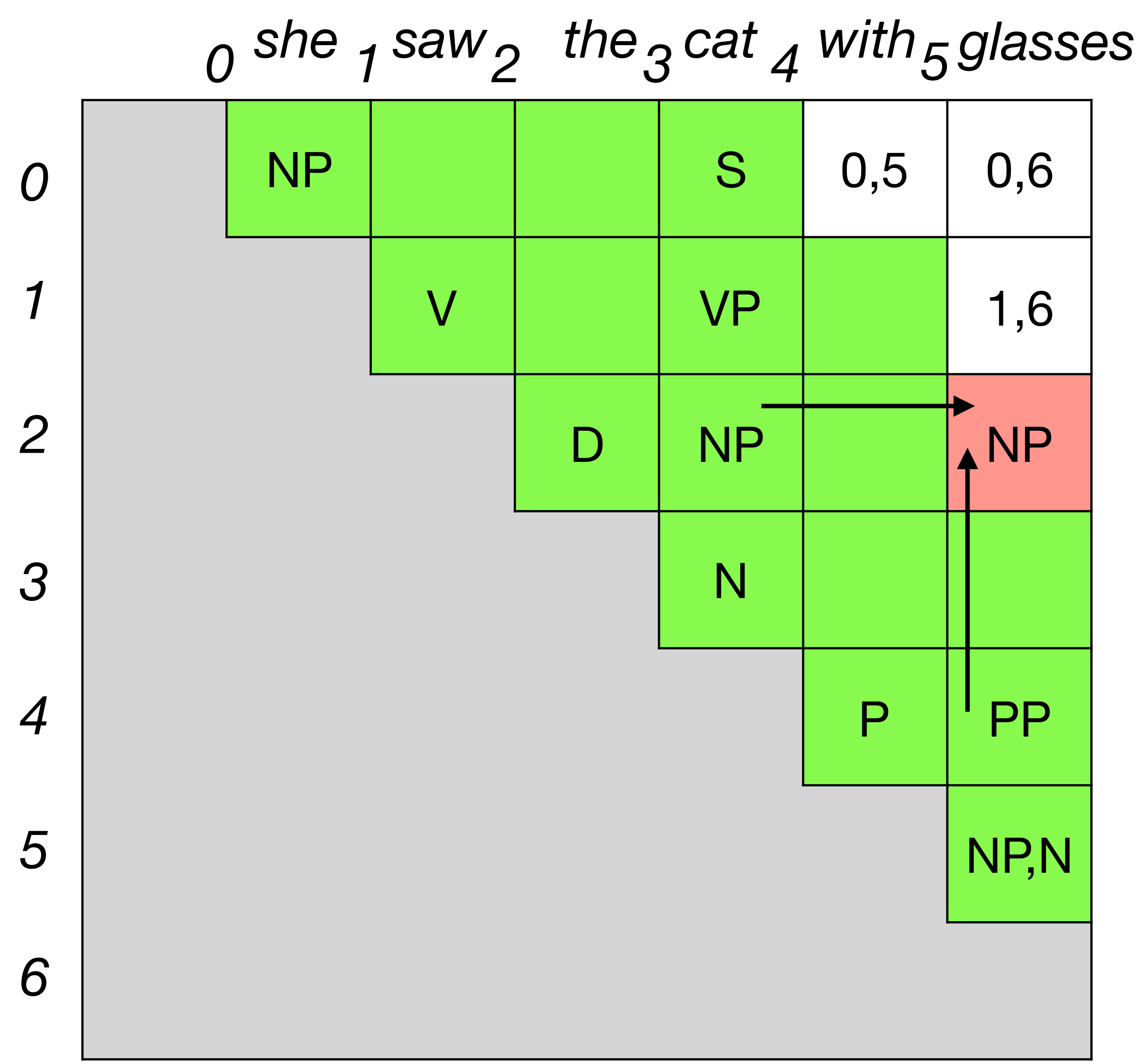
  for $k=i+1 \ldots j\text{-}1$:

  ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=4$

$i=2, k=4, j=6$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

  $j = i+length$

  for $k=i+1\ldots j-1$:

    ....

S	→ NP VP		NP	→ she	
VP	→ V NP		NP	→ glasses	
VP	→ VP PP		D	→ the	
PP	→ P NP		N	→ cat	
NP	→ D N		N	→ glasses	
NP	→ NP PP		V	→ saw	
			P	→ with	

$length=4$

$i=2, k=5, j=6$

$_0$ *she* $_1$ *saw* $_2$ *the* $_3$ *cat* $_4$ *with* $_5$ *glasses*

	1	2	3	4	5	6
0	NP			S	0,5	0,6
1		V		VP		1,6
2			D	NP		NP
3				N		
4					P	PP
5						NP,N
6						

CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

    $j = i+length$

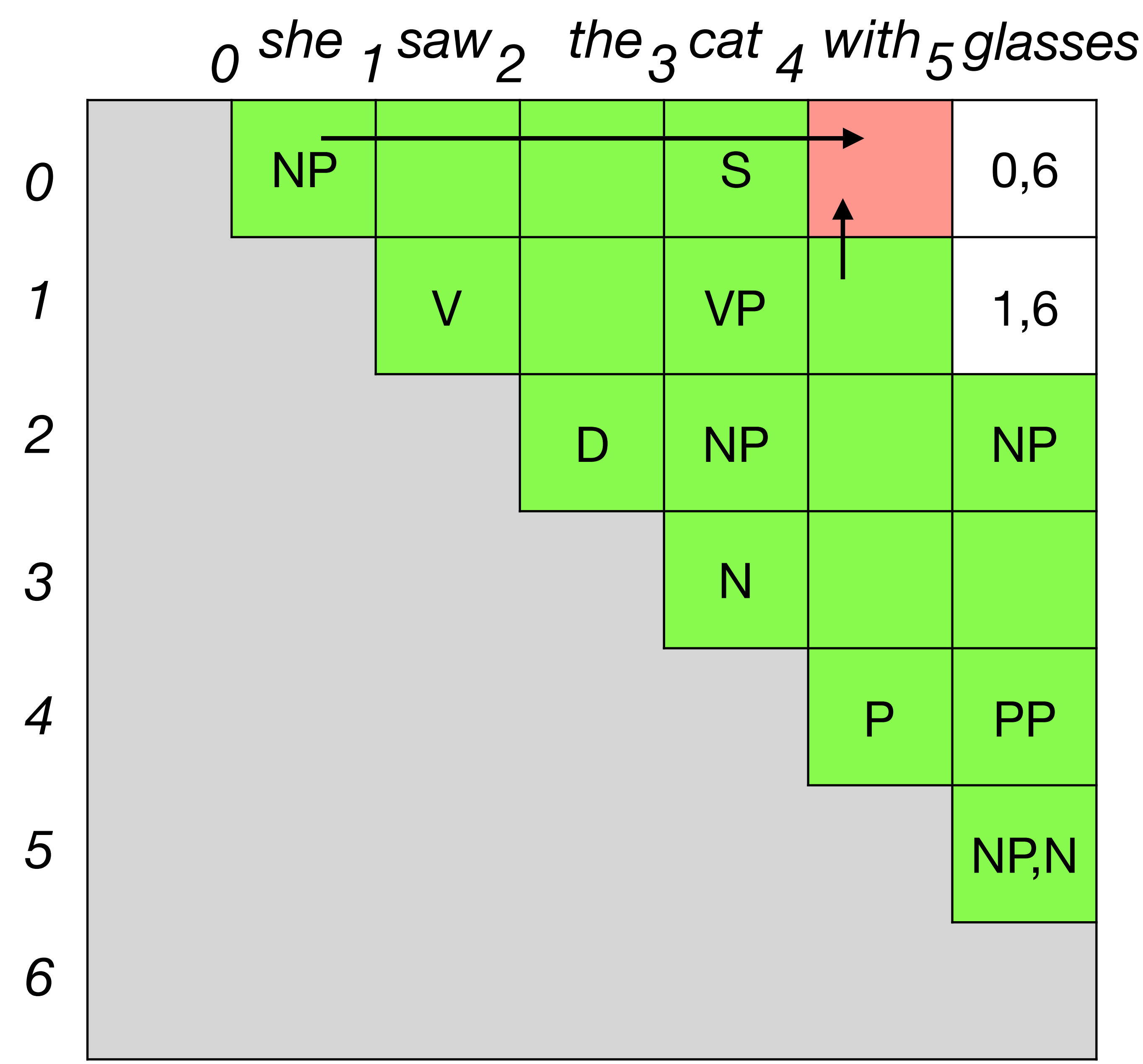
    for $k=i+1\ldots j\text{-}1$:

        ....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=5$

$i=0,k=1,j=5$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

    $j = i+length$

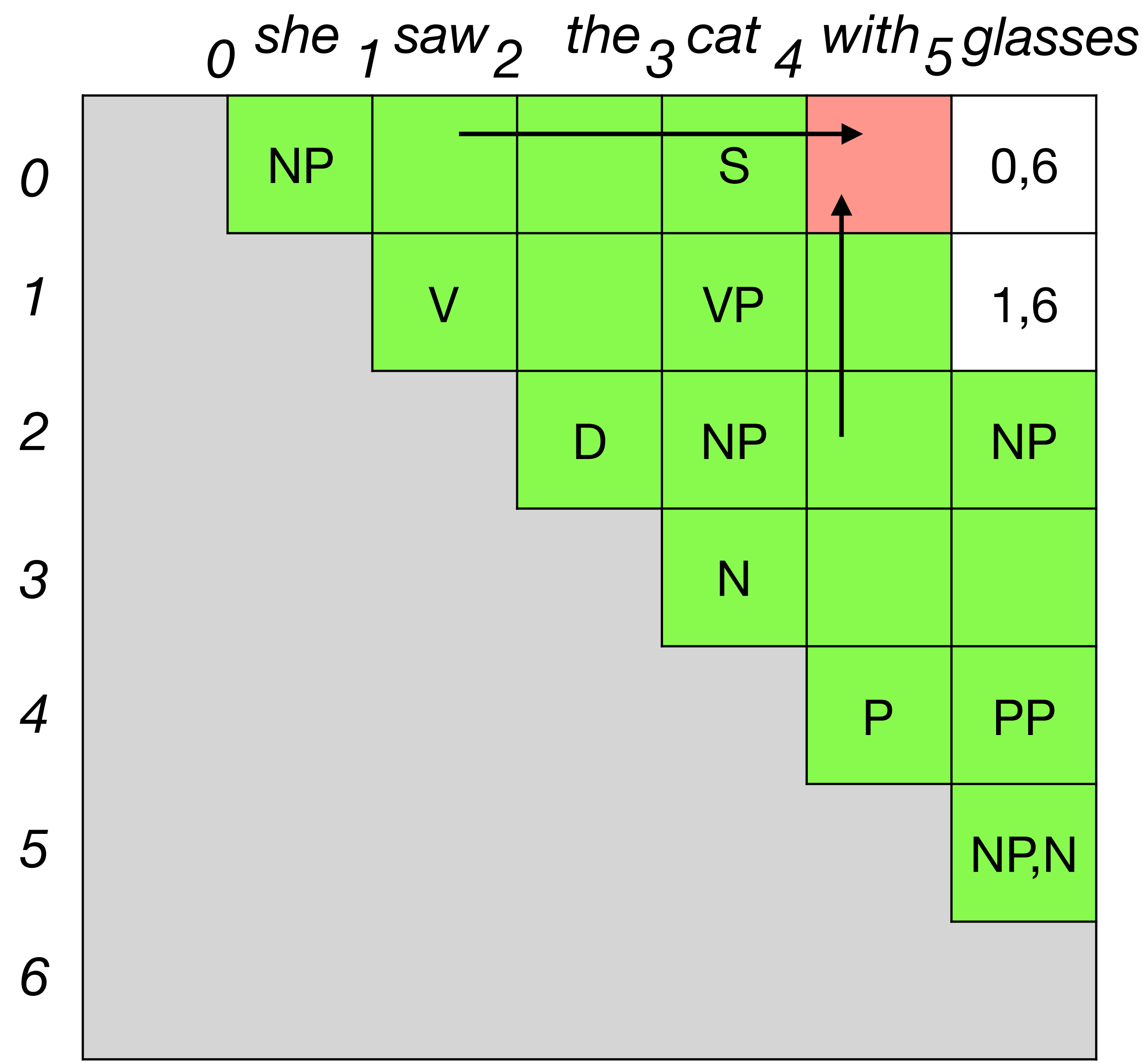
    for $k=i+1\ldots j\text{-}1$:

        ....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=5$

$i=0,k=2,j=5$



CKY Algorithm

for $i=0...(n\text{-}length)$:

  $j = i+length$

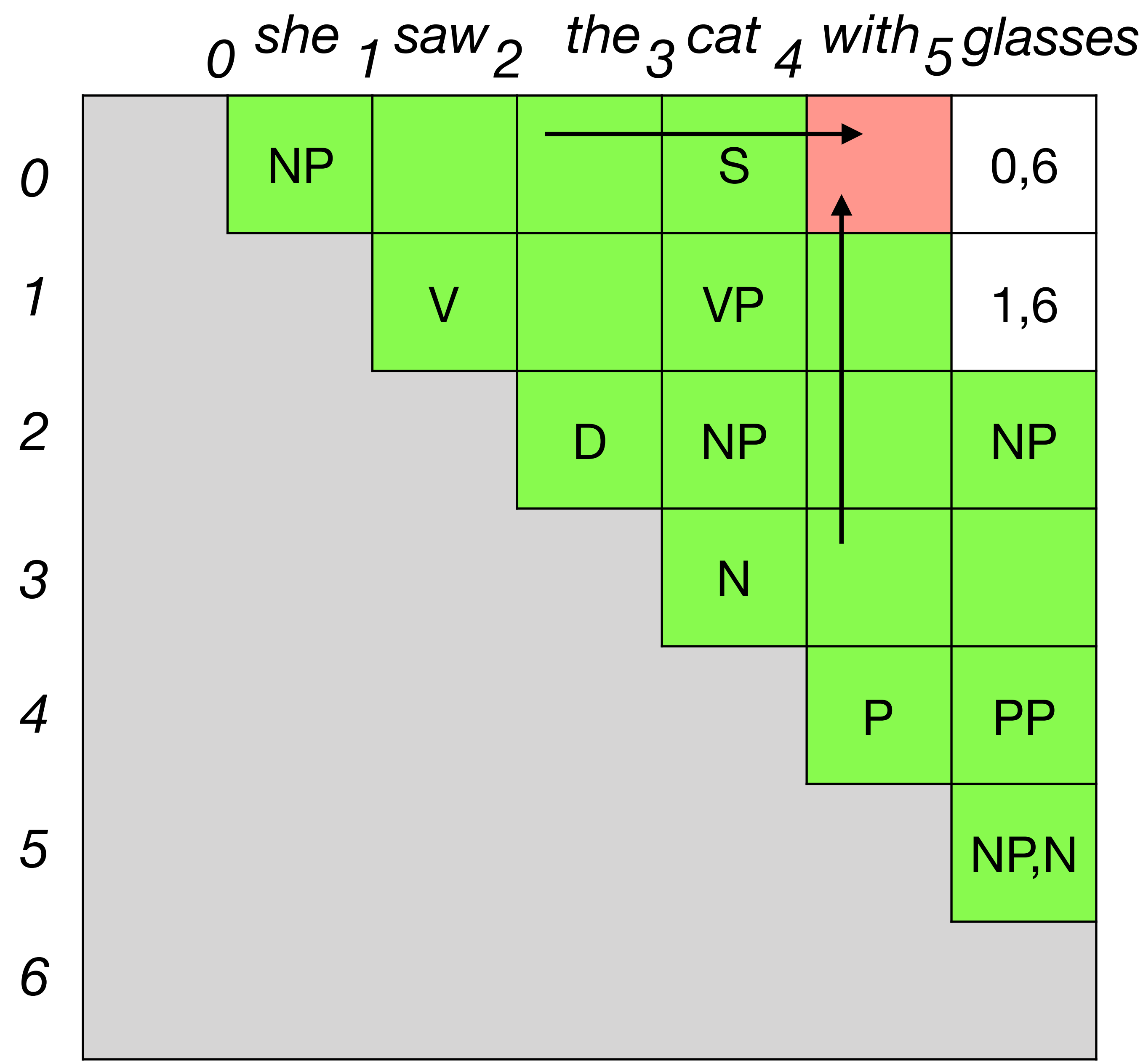
  for $k=i+1...j\text{-}1$:

   ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=5$

$i=0,k=3,j=5$



CKY Algorithm

for *i=0…(n-length):*
  j = i+length
  for *k=i+1…j-1:*
    ….

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

length=5

i=0,k=4,j=5

0 she 1 saw 2 the 3 cat 4 with 5 glasses

0	NP			S		0,6
1		V		VP		1,6
2			D	NP		NP
3				N		
4					P	PP
5						NP,N
6						

CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

  $j = i+length$

  for $k=i+1 \ldots j\text{-}1$:

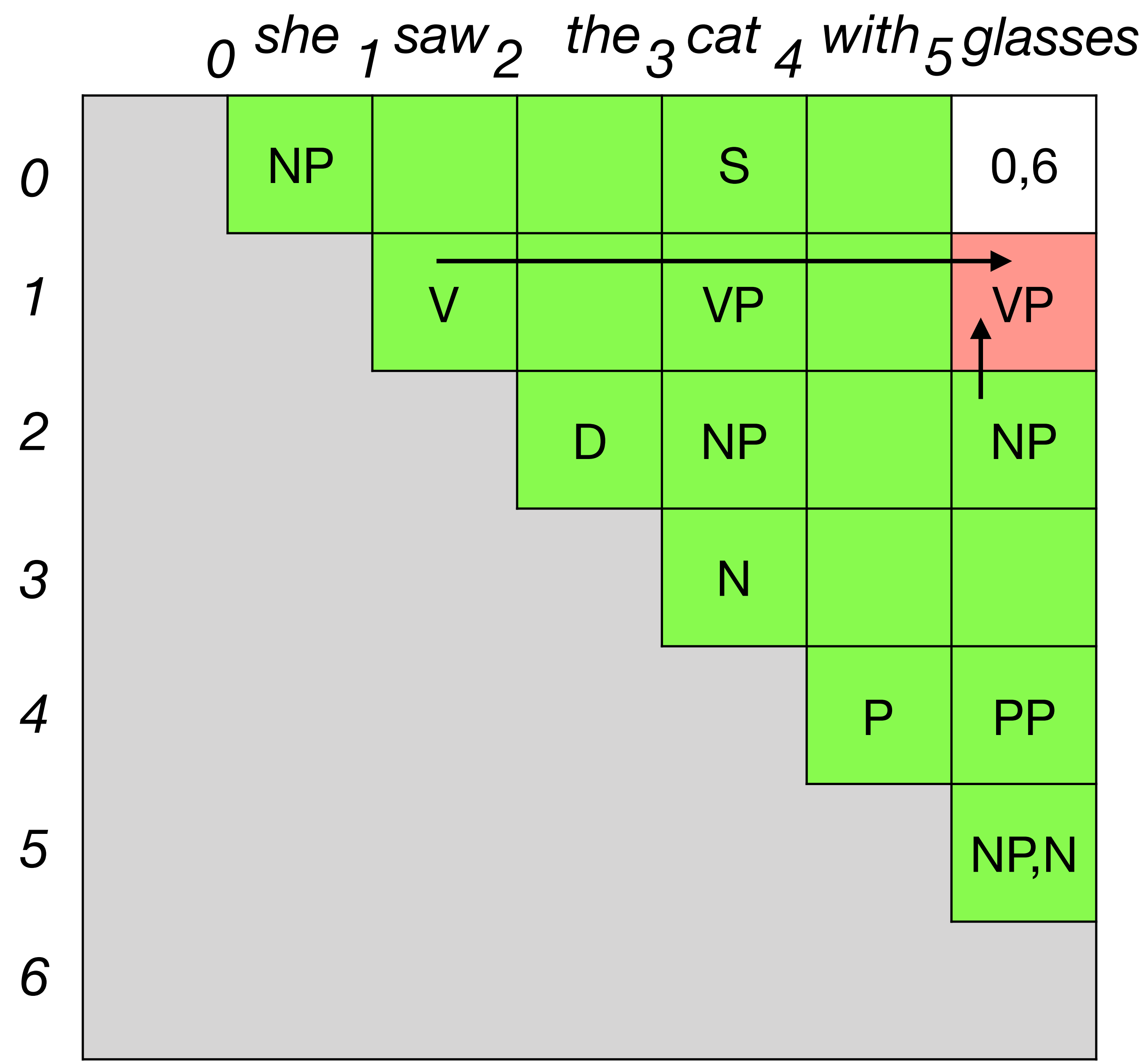
  ….

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

NP → she
NP → glasses
D → the
N → cat
N → glasses
V → saw
P → with

$length=5$

$i=1, k=2, j=6$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

$\quad j = i+length$

$\quad$ for $k=i+1\ldots j\text{-}1$:

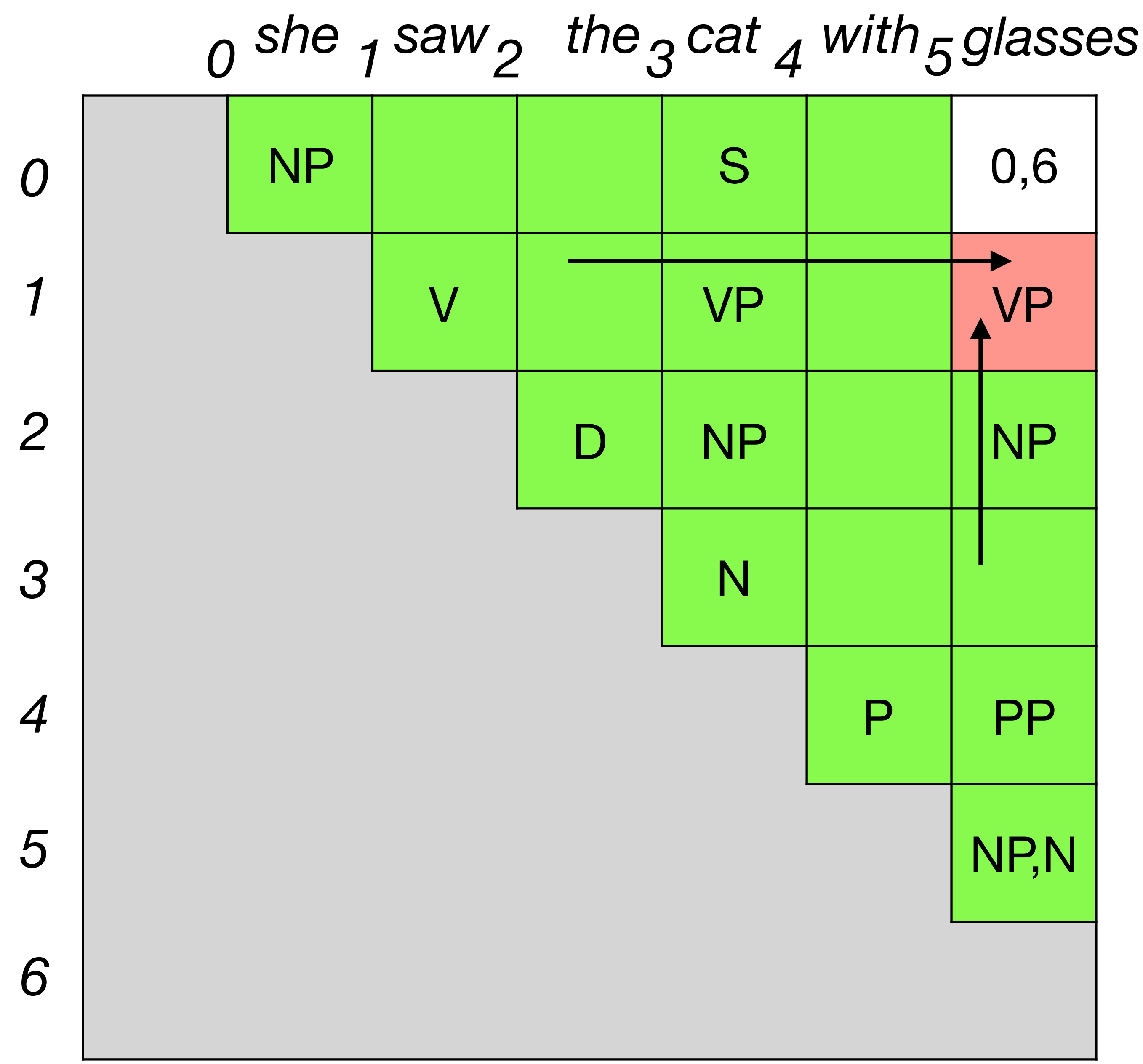
$\quad\quad\ldots.$

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

NP → she
NP → glasses
D → the
N → cat
N → glasses
V → saw
P → with

$length=5$

$i=1,k=3,j=6$



CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

$j = i+length$

for $k=i+1 \ldots j-1$:

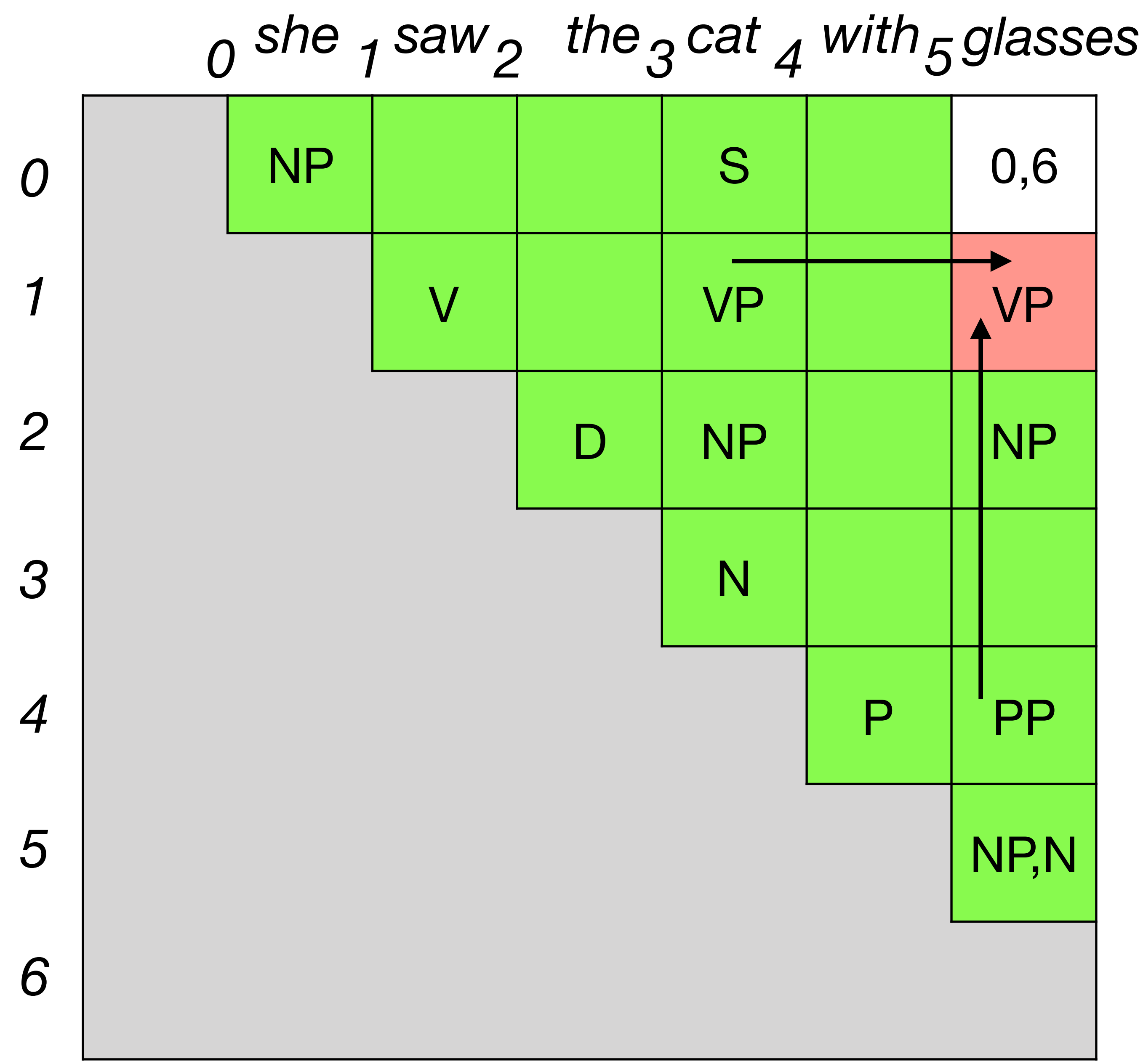
....

S → NP VP		NP → she	
VP → V NP		NP → glasses	
VP → VP PP		D → the	
PP → P NP		N → cat	
NP → D N		N → glasses	
NP → NP PP		V → saw	
		P → with	

! We can build VP over [1,6] in two ways!

$length=5$

$i=1, k=4, j=6$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

$\quad j = i+length$

$\quad$ for $k=i+1\ldots j\text{-}1$:

$\quad\quad$ ….

S	→ NP VP		NP	→ she
VP	→ V NP		NP	→ glasses
VP	→ VP PP		D	→ the
PP	→ P NP		N	→ cat
NP	→ D N		N	→ glasses
NP	→ NP PP		V	→ saw
			P	→ with

$length=5$

$i=0,k=1,j=6$

0 she 1 saw 2 the 3 cat 4 with 5 glasses

	0	1	2	3	4	5	6
0		NP			S		S
1			V		VP		VP
2				D	NP		NP
3					N		
4						P	PP
5							NP,N
6							

CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

$\quad j = i+length$

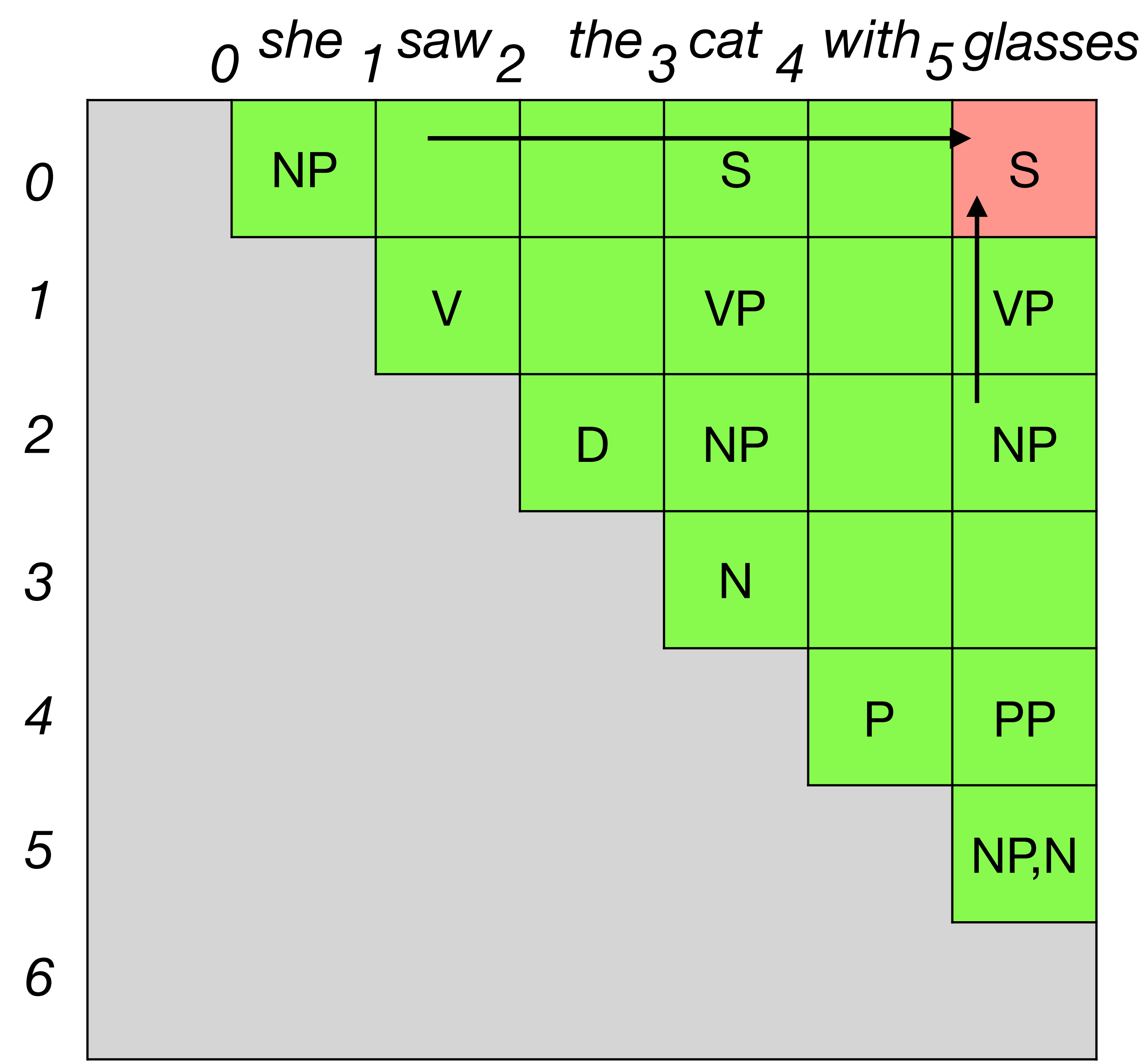
$\quad$ for $k=i+1\ldots j\text{-}1$:

$\quad\quad\ldots.$

S → NP VP		NP → she	
VP → V NP		NP → glasses	
VP → VP PP		D → the	
PP → P NP		N → cat	
NP → D N		N → glasses	
NP → NP PP		V → saw	
		P → with	

$length=5$

$i=0,k=2,j=6$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

$\quad j = i+length$

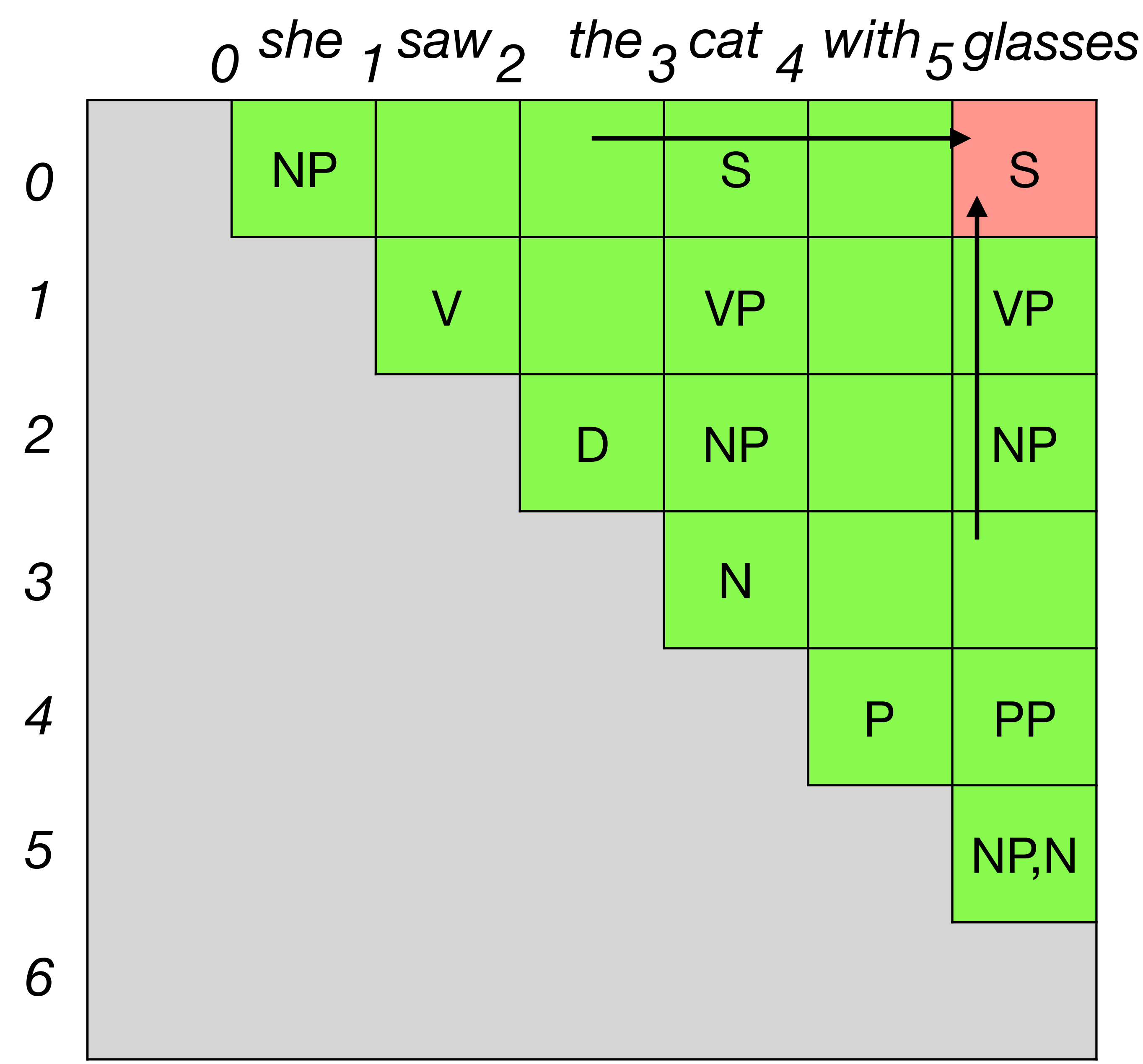
$\quad$ for $k=i+1\ldots j\text{-}1$:

$\quad\quad\ldots.$

S	→	NP VP	NP	→	she
VP	→	V NP	NP	→	glasses
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	glasses
NP	→	NP PP	V	→	saw
			P	→	with

$length=5$

$i=0,k=3,j=6$



CKY Algorithm

for $i=0 \ldots (n\text{-}length)$:

  $j = i+length$

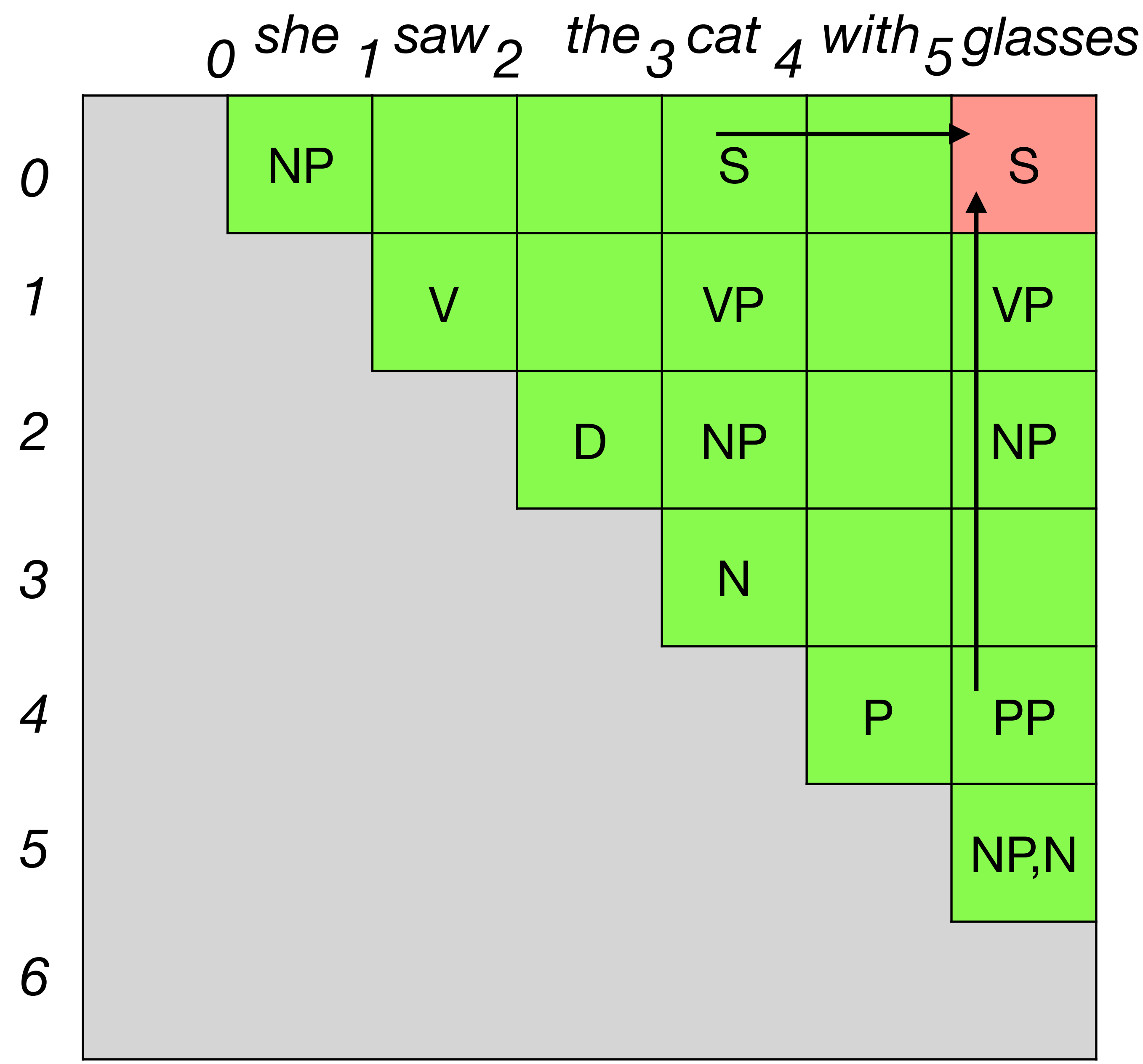
  for $k=i+1 \ldots j\text{-}1$:

  ….

S	→	NP VP	NP	→	she
VP	→	V NP	NP	→	glasses
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	glasses
NP	→	NP PP	V	→	saw
			P	→	with

$length=5$

$i=0, k=4, j=6$



CKY Algorithm

for $i=0\ldots(n\text{-}length)$:

    $j = i+length$

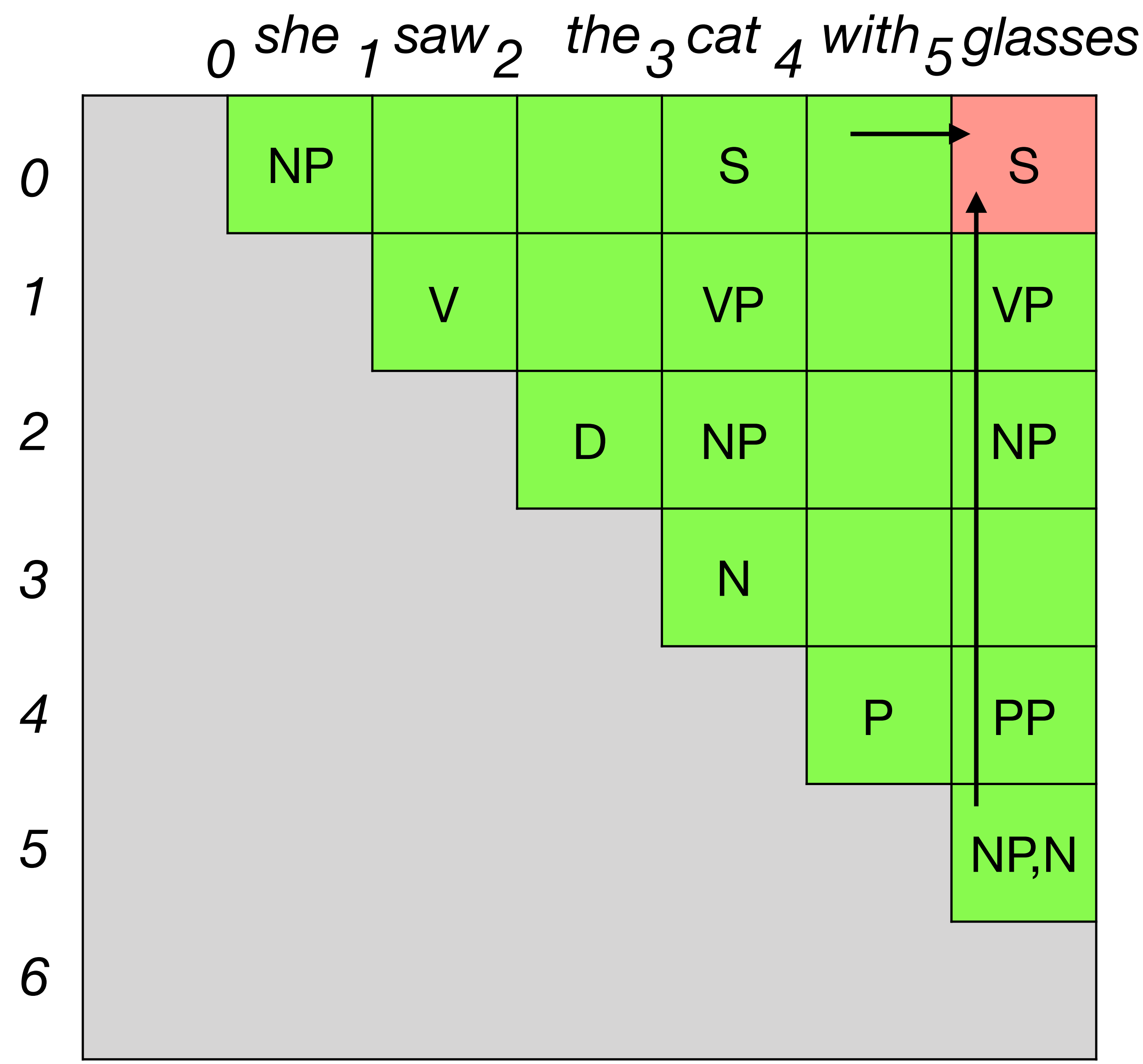
    for $k=i+1\ldots j\text{-}1$:

        ....

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with

$length=5$

$i=0,k=5,j=6$



CKY Runtime

- **Input:** Grammar $G=(N, \Sigma, R, S)$, input string s of length n.

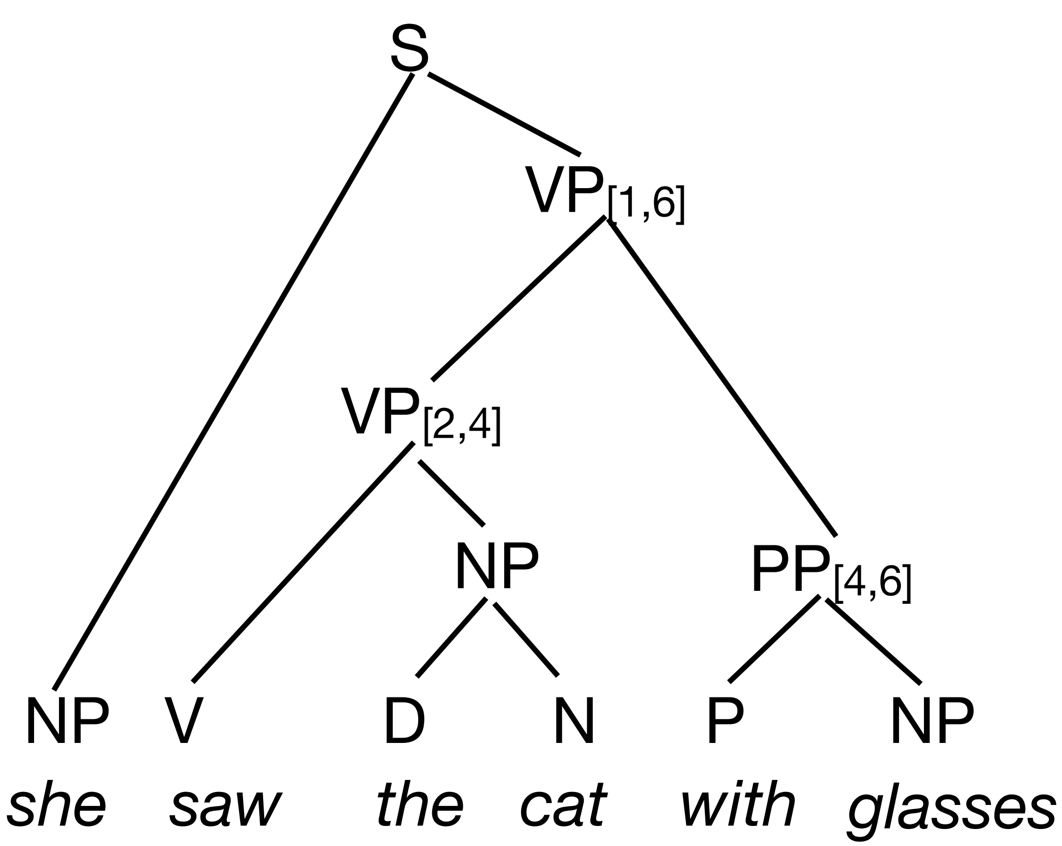
- for i=0…n-1: O(N x |R|)
  π[i, i+1] = {A | A → s[i] }

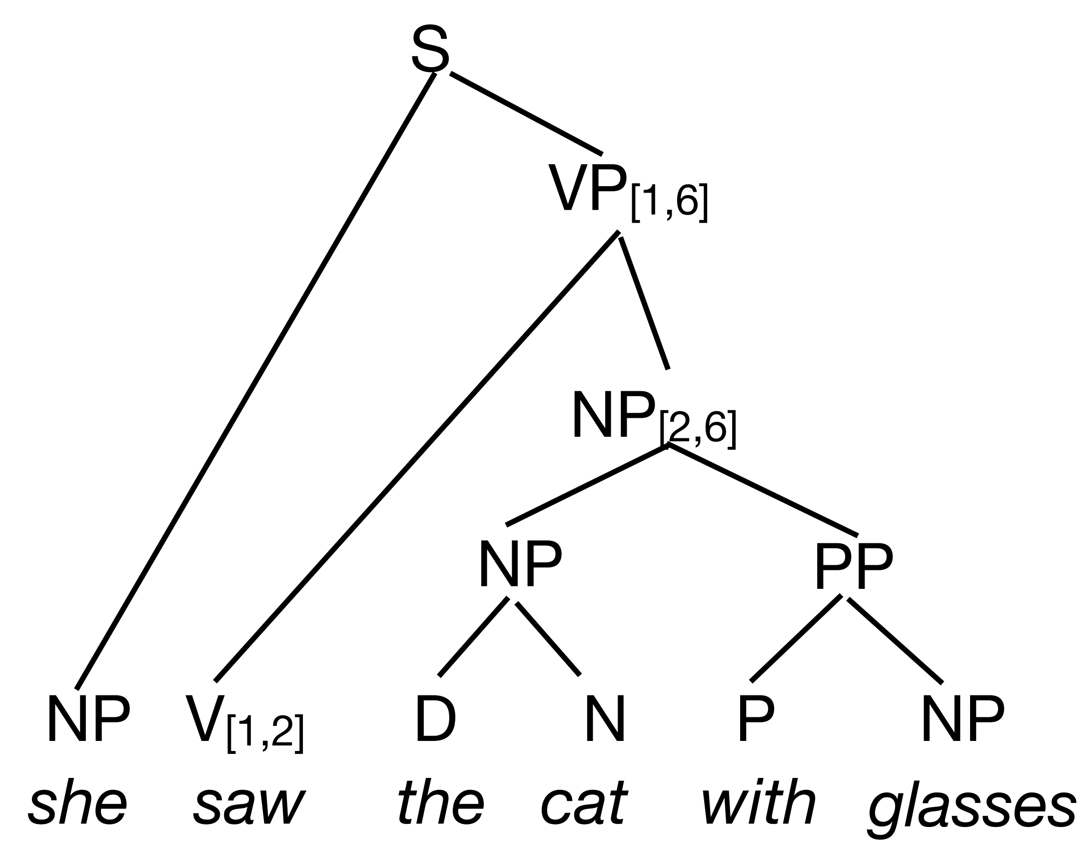
- for $length=2 \ldots n$: O(N)
  for $i=0 \ldots (n\text{-}length)$: O(N) Total : O($N^3$ x |R|)
  $j = i+length$ O(N)
  for $k=i+1 \ldots j\text{-}1$:
  $$M = \{A|A \rightarrow B\,C \in R \text{ and } B \in \pi[i,k] \text{ and } C \in \pi[k,j]\}$$
  $$\pi[i,j] = \pi[i,j] \cup M$$

- if $S \in \pi[0, i+1]$ return True, otherwise False

Syntactic Ambiguity

S → NP VP	NP → she
VP → V NP	NP → glasses
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → glasses
NP → NP PP	V → saw
	P → with





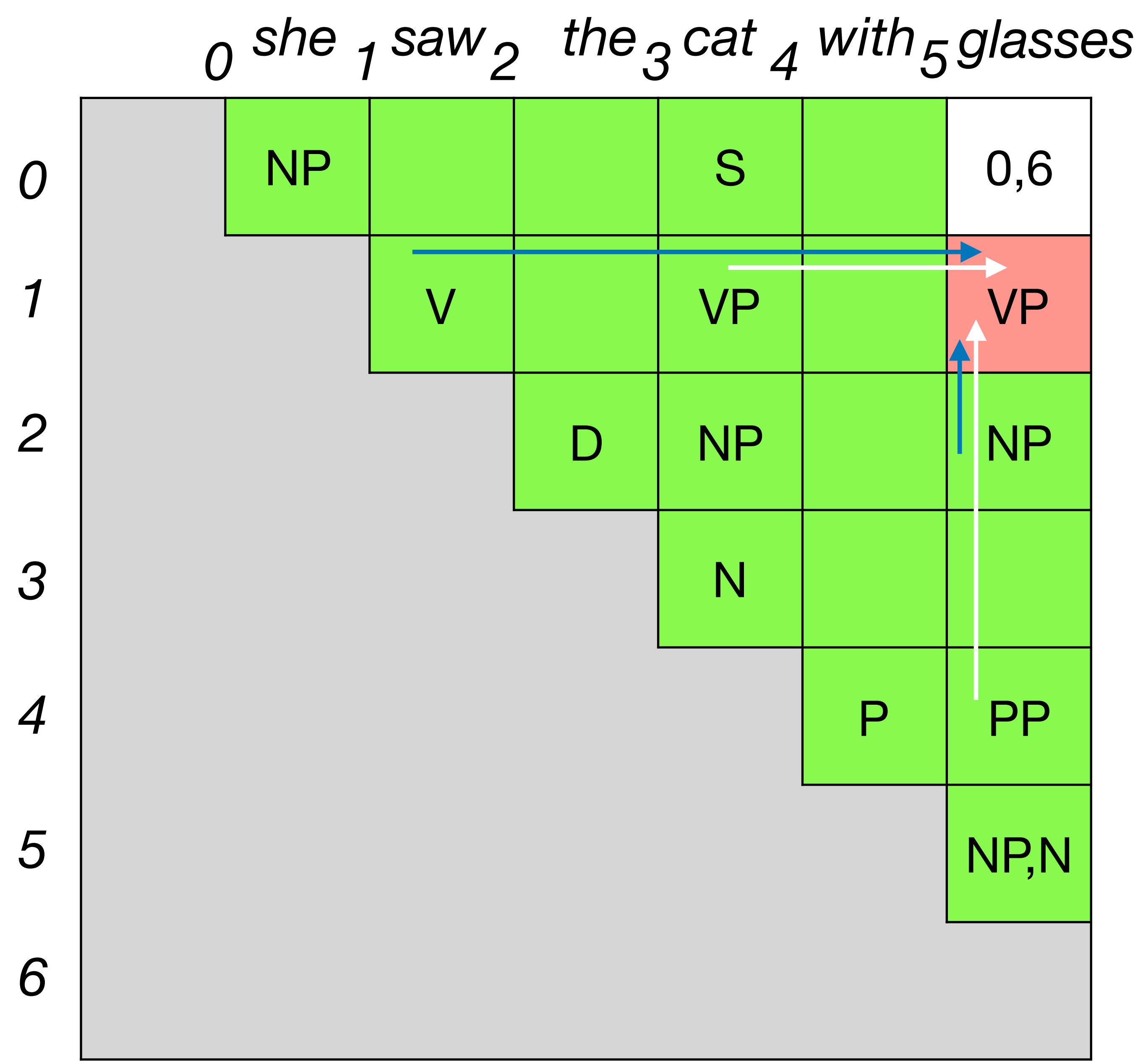
Backpointers

- The CKY algorithm presented so far determines if a sentence is recognized by a grammar.

- Also want to retrieve the parse trees!

- Instead of a set of nonterminals, store a list of instantiated rules and backpointers.

$$\left\{ \begin{array}{l} VP_{[1,6]} \rightarrow V_{[1,2]} \;\; NP_{[2,6]} \\ VP_{[1,6]} \rightarrow VP_{[1,4]} \; PP_{[4,6]} \end{array} \right\}$$



Retrieving Parse-Trees

- Start at the [0,n] entry and recursively follow the backpointers. Return a set of of subtrees from the recursion.
- How long does it take to retrieve all parse trees?
  - Worst case, there are exponentially many trees. So retrieving all of them is exponential.
- However: We can retrieve the *k* highest-scoring trees in polynomial time (next week).
- Retrieving ANY single parse tree takes $O(N^2)$.