

Remembering subresults (Part I): Well-formed substring tables

Detmar Meurers: Intro to Computational Linguistics I
OSU, LING 684.01

Problem: Inefficiency of recomputing subresults

Two example sentences and their potential analysis:

- (1) He [gave [the young cat] [to Bill]].
- (2) He [gave [the young cat] [some milk]].

The corresponding grammar rules:

vp ----> [v_ditrans, np, pp_to].
vp ----> [v_ditrans, np, np].

Solution: Memoization

- Store intermediate results:
 - a) completely analyzed constituents:
well-formed substring table or **(passive) chart**
 - b) partial and complete analyses:
(active) chart
- All intermediate results need to be stored for completeness.
- All possible solutions are explored in parallel.

CFG Parsing: The Cocke Younger Kasami Algorithm

- Grammar has to be in Chomsky Normal Form (CNF), only
 - RHS with a single terminal: $A \rightarrow a$
 - RHS with two non-terminals: $A \rightarrow BC$
 - no ϵ rules ($A \rightarrow \epsilon$)
- A representation of the string showing positions and word indices:

$\cdot_0 w_1 \cdot_1 w_2 \cdot_2 w_3 \cdot_3 w_4 \cdot_4 w_5 \cdot_5 w_6 \cdot_6$

For example: \cdot_0 the \cdot_1 young \cdot_2 boy \cdot_3 saw \cdot_4 the \cdot_5 dragon \cdot_6

The well-formed substring table (= passive chart)

- The well-formed substring table, henceforth (passive) chart, for a string of length n is an $n \times n$ matrix.
- The field (i, j) of the chart encodes the set of all categories of constituents that start at position i and end at position j , i.e.
$$\text{chart}(i,j) = \{A \mid A \Rightarrow^* w_{i+1} \dots w_j\}$$
- The matrix is triangular since no constituent ends before it starts.

Coverage Represented in the Chart

An input sentence with 6 words:

$$\cdot_0 w_1 \cdot_1 w_2 \cdot_2 w_3 \cdot_3 w_4 \cdot_4 w_5 \cdot_5 w_6 \cdot_6$$

Coverage represented in the chart:

		TO:					
		1	2	3	4	5	6
FROM:	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

Example for Coverage Represented in Chart

Example sentence:

·₀ the ·₁ young ·₂ boy ·₃ saw ·₄ the ·₅ dragon ·₆

Coverage represented in chart:

	1	2	3	4	5	6
0	the	the young	the young boy	the young boy saw	the young boy saw the	the young boy saw the dragon
1		young	young boy	young boy saw	young boy saw the	young boy saw the dragon
2			boy	boy saw	boy saw the	boy saw the dragon
3				saw	saw the	saw the dragon
4					the	the dragon
5						dragon

An Example for a Filled-in Chart

Input sentence:

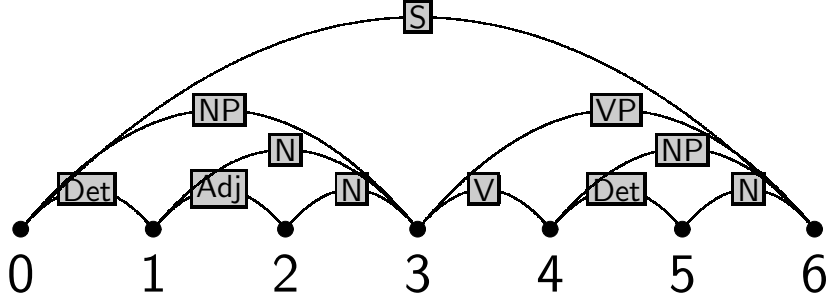
·₀ the ·₁ young ·₂ boy ·₃ saw ·₄ the ·₅ dragon ·₆

Chart:

	1	2	3	4	5	6
0	{Det}	{ }	{NP}	{ }	{ }	{S}
1		{Adj}	{N}	{ }	{ }	{ }
2			{N}	{ }	{ }	{ }
3				{V, N}	{ }	{VP}
4					{Det}	{NP}
5						{N}

Grammar:

- S → NP VP
- VP → Vt NP
- NP → Det N
- N → Adj N
- Vt → saw
- Det → the
- Det → a
- N → dragon
- N → boy
- N → saw
- Adj → young



Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to  $\text{length}(\text{string})$   
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to  $\text{length}(\text{string})$   
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to  $\text{length}(\text{string})$   
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to  $0$   
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

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

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6			
1		2	5			
2			4			
3				7		
4						
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```


Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6			
1		2	5			
2			4	8		
3				7		
4						
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6			
1		2	5	9		
2			4	8		
3				7		
4						
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10		
1		2	5	9		
2			4	8		
3				7		
4						
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10		
1		2	5	9		
2			4	8		
3				7		
4					11	
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10		
1		2	5	9		
2			4	8		
3				7	12	
4					11	
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10		
1		2	5	9		
2			4	8	13	
3				7	12	
4					11	
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10		
1		2	5	9	14	
2			4	8	13	
3				7	12	
4					11	
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	
1		2	5	9	14	
2			4	8	13	
3				7	12	
4					11	
5						

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```


Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	
1		2	5	9	14	
2			4	8	13	
3				7	12	
4					11	
5						16

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	
1		2	5	9	14	
2			4	8	13	
3				7	12	
4					11	17
5						16

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	
1		2	5	9	14	
2			4	8	13	
3				7	12	18
4					11	17
5						16

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	
1		2	5	9	14	
2			4	8	13	19
3				7	12	18
4					11	17
5						16

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	
1		2	5	9	14	20
2			4	8	13	19
3				7	12	18
4					11	17
5						16

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

Filling in the Chart

- It is important to fill in the chart systematically.
- We build all constituents that end at a certain point before we build constituents that end at a later point.

	1	2	3	4	5	6
0	1	3	6	10	15	21
1		2	5	9	14	20
2			4	8	13	19
3				7	12	18
4					11	17
5						16

```
for  $j := 1$  to length(string)  
  lexical_chart_fill( $j - 1, j$ )  
  for  $i := j - 2$  down to 0  
    syntactic_chart_fill( $i, j$ )
```

lexical_chart_fill(j-1,j)

- Idea: Lexical lookup. Fill the field $(j - 1, j)$ in the chart with the preterminal category dominating word j .
- Realized as:

$$\text{chart}(j - 1, j) := \{X \mid X \rightarrow \text{word}_j \in P\}$$

syntactic_chart_fill(i,j)

- Idea: Perform all reduction step using syntactic rules such that the reduced symbol covers the string from i to j .

- Realized as:
$$chart(i, j) = \left\{ A \mid \begin{array}{l} A \rightarrow BC \in P, \\ i < k < j, \\ B \in chart(i, k), \\ C \in chart(k, j) \end{array} \right\}$$

- Explicit loops over every possible value of k and every context free rule:

$chart(i, j) := \{\}$.

for $k := i + 1$ to $j - 1$

 for every $A \rightarrow BC \in P$

 if $B \in chart(i, k)$ and $C \in chart(k, j)$ then

$chart(i, j) := chart(i, j) \cup \{A\}$.

The Complete CYK Algorithm

Input: start category S and input *string*

$n := \text{length}(\textit{string})$

for $j := 1$ to n

$\textit{chart}(j - 1, j) := \{X \mid X \rightarrow \textit{word}_j \in P\}$

for $i := j - 2$ down to 0

$\textit{chart}(i, j) := \{\}$

for $k := i + 1$ to $j - 1$

for every $A \rightarrow BC \in P$

if $B \in \textit{chart}(i, k)$ and $C \in \textit{chart}(k, j)$ then

$\textit{chart}(i, j) := \textit{chart}(i, j) \cup \{A\}$

Output: if $S \in \textit{chart}(0, n)$ then accept else reject

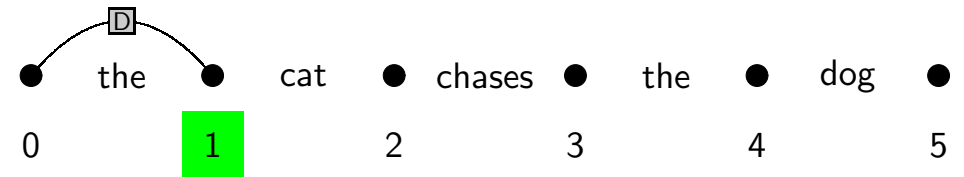
Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

Lexical Entry: *the*

($j = 1$, field chart(0,1))

	1	2	3	4	5
0	d				
1					
2					
3					
4					



Example Application of the CYK Algorithm

$s \rightarrow np\ vp$

$np \rightarrow d\ n$

$vp \rightarrow v\ np$

$d \rightarrow the$

$n \rightarrow dog$

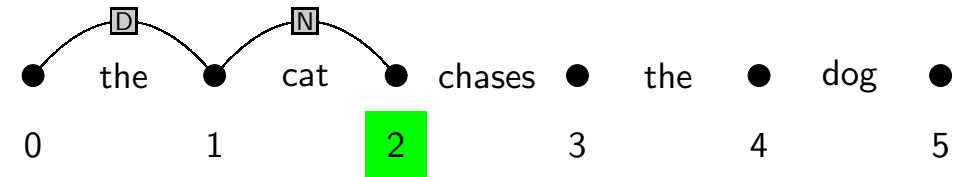
$n \rightarrow cat$

$v \rightarrow chases$

Lexical Entry: *cat*

($j = 2$, field chart(1,2))

	1	2	3	4	5
0	d				
1		n			
2					
3					
4					

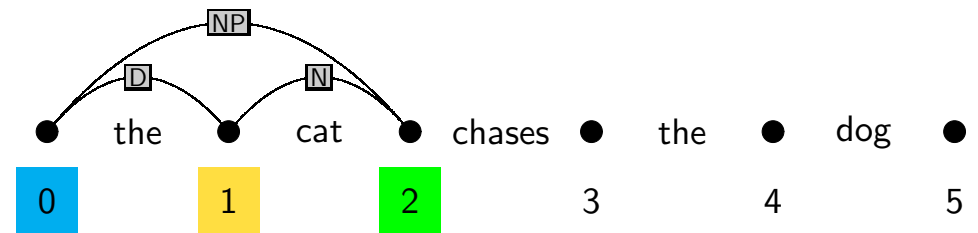


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

$j = 2$
 $i = 0$
 $k = 1$

	1	2	3	4	5
0	d	np			
1		n			
2					
3					
4					



Example Application of the CYK Algorithm

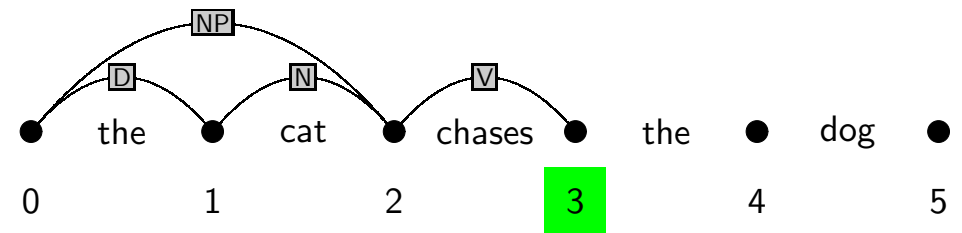
$s \rightarrow np\ vp$
 $np \rightarrow d\ n$
 $vp \rightarrow v\ np$

$d \rightarrow the$
 $n \rightarrow dog$
 $n \rightarrow cat$
 $v \rightarrow chases$

Lexical Entry: *chases*

($j = 3$, field chart(2,3))

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					

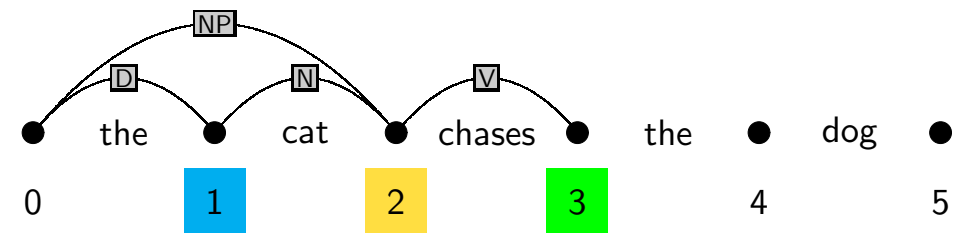


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					

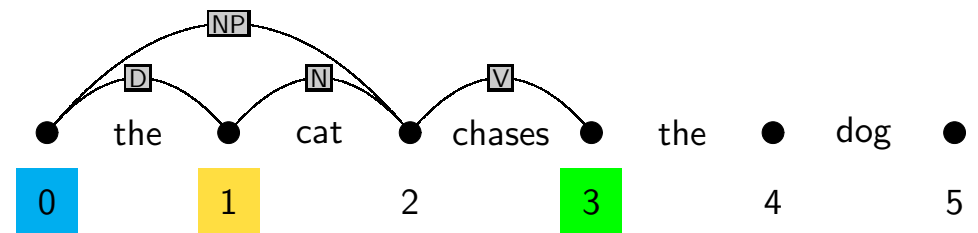


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					

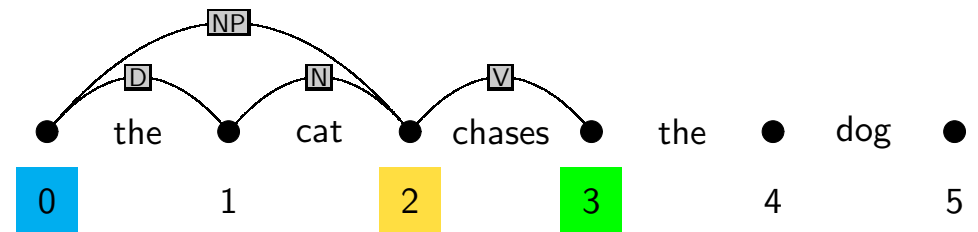


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					



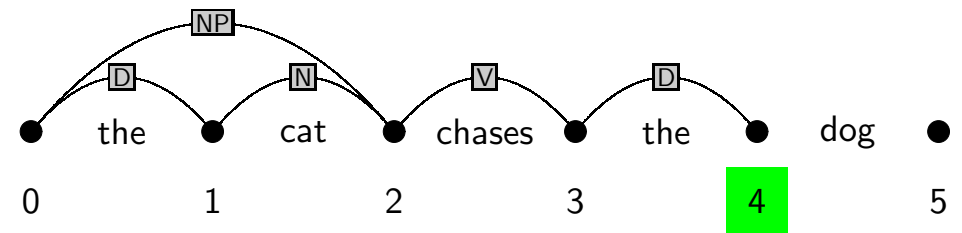
Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow \text{the}$
 $np \rightarrow d\ n$ $n \rightarrow \text{dog}$
 $vp \rightarrow v\ np$ $n \rightarrow \text{cat}$
 $v \rightarrow \text{chases}$

Lexical Entry: *the*

($j = 4$, field chart(3,4))

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					

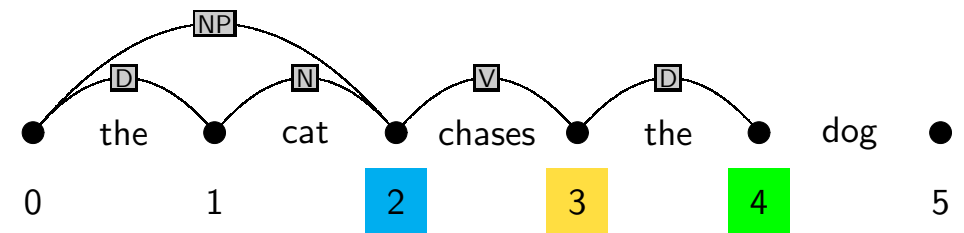


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

$j = 4$
 $i = 2$
 $k = 3$

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					

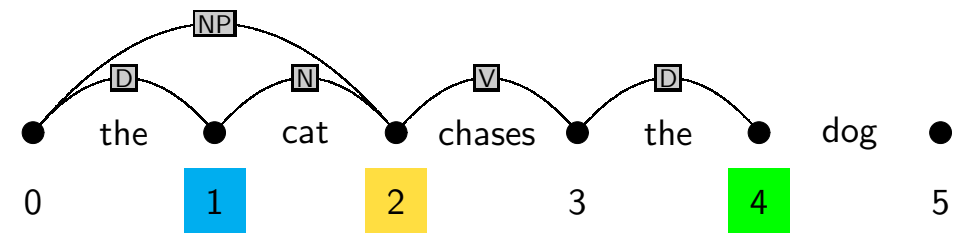


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

$j = 4$
 $i = 1$
 $k = 2$

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					

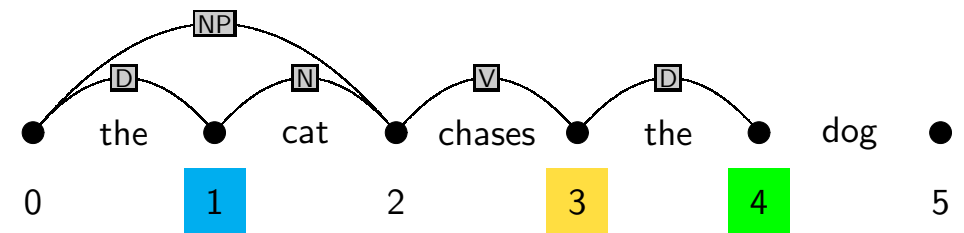


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n	<input type="checkbox"/>	<input type="checkbox"/>	
2			v		
3				d	
4					

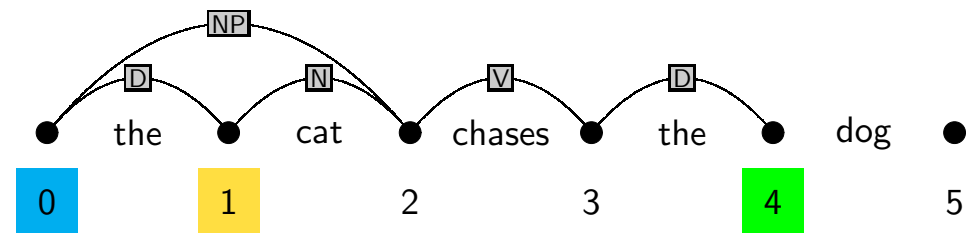


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					

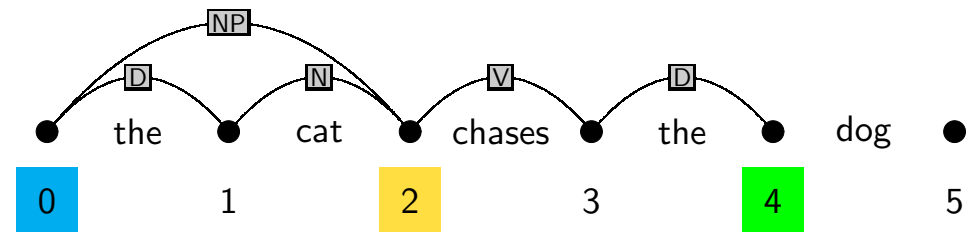


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					

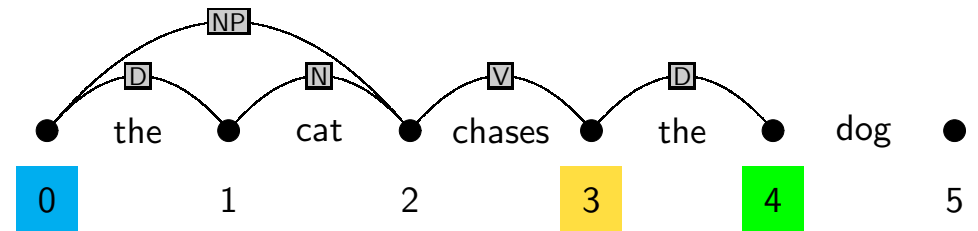


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np	<input type="checkbox"/>	<input type="checkbox"/>	
1		n			
2			v		
3				d	
4					



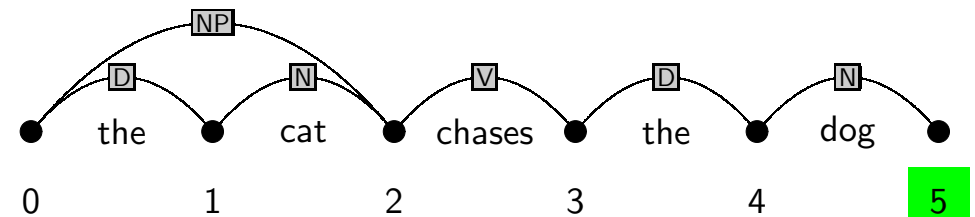
Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

Lexical Entry: *dog*

($j = 5$, field chart(4,5))

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					n



Example Application of the CYK Algorithm

$s \rightarrow np\ vp$

$np \rightarrow d\ n$

$vp \rightarrow v\ np$

$d \rightarrow the$

$n \rightarrow dog$

$n \rightarrow cat$

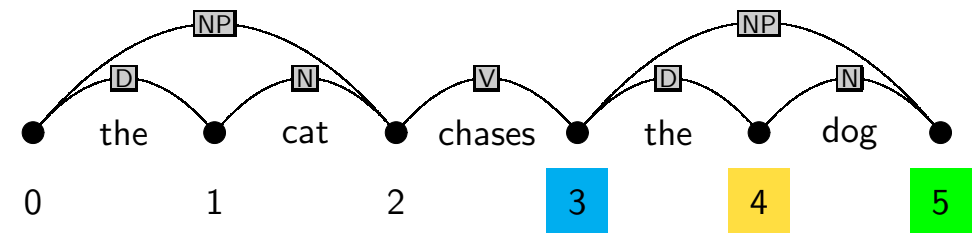
$v \rightarrow chases$

$j = 5$

$i = 3$

$k = 4$

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	np
4					n

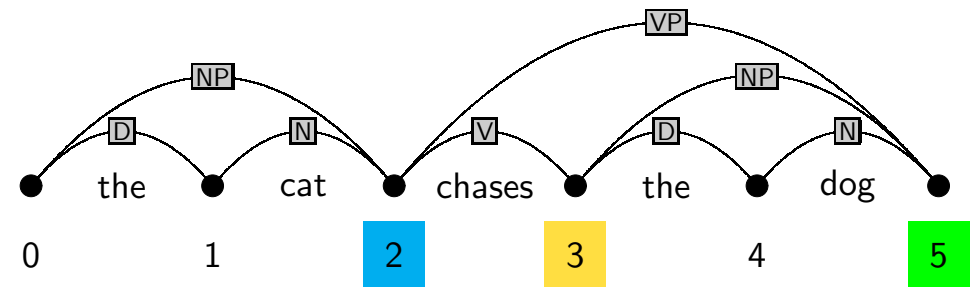


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

$j = 5$
 $i = 2$
 $k = 3$

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n

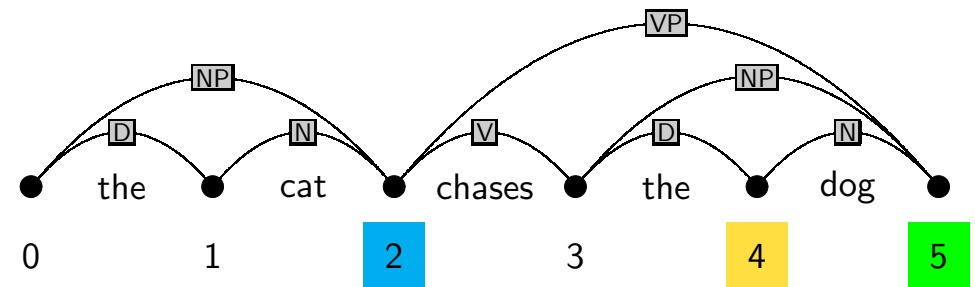


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n

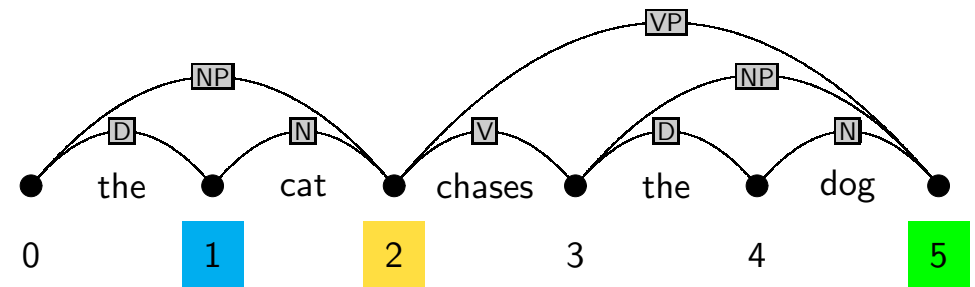


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n

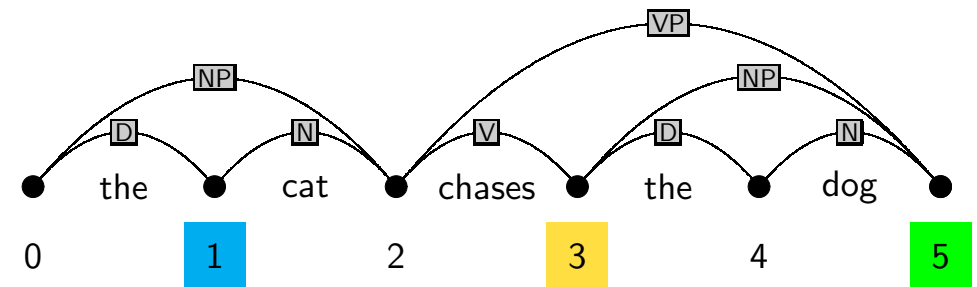


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

$j = 5$
 $i = 1$
 $k = 3$

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n

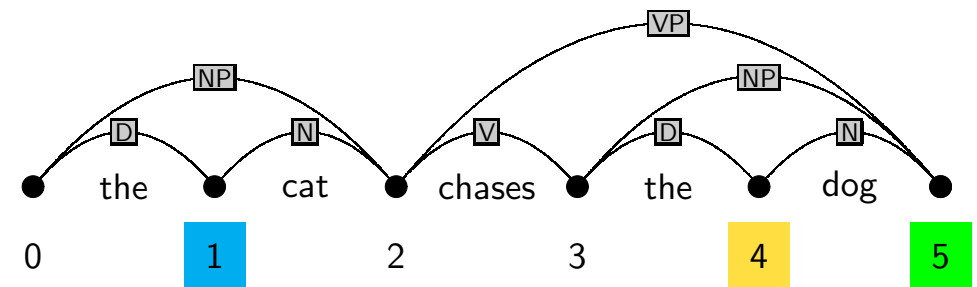


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n		<input type="checkbox"/>	<input type="checkbox"/>
2			v		vp
3				d	np
4					<input type="checkbox"/> n

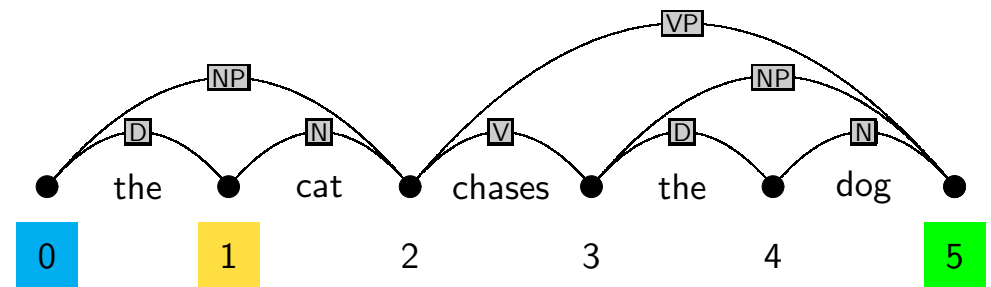


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n

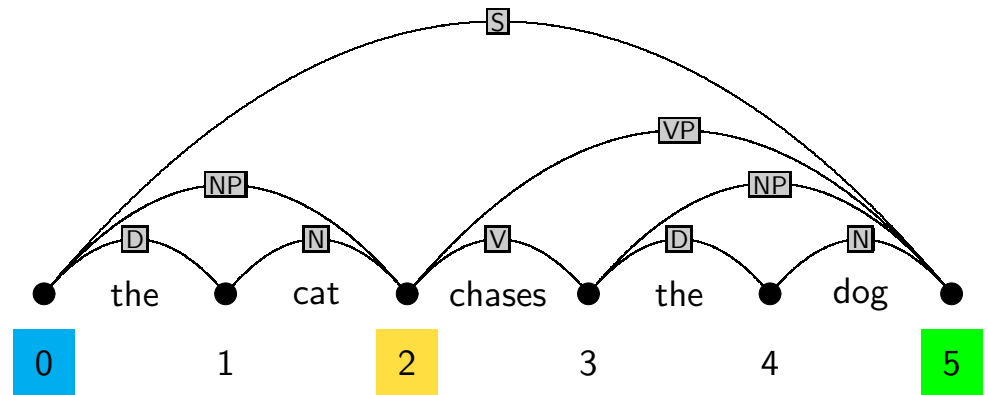


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			s
1		n			
2			v		vp
3				d	np
4					n

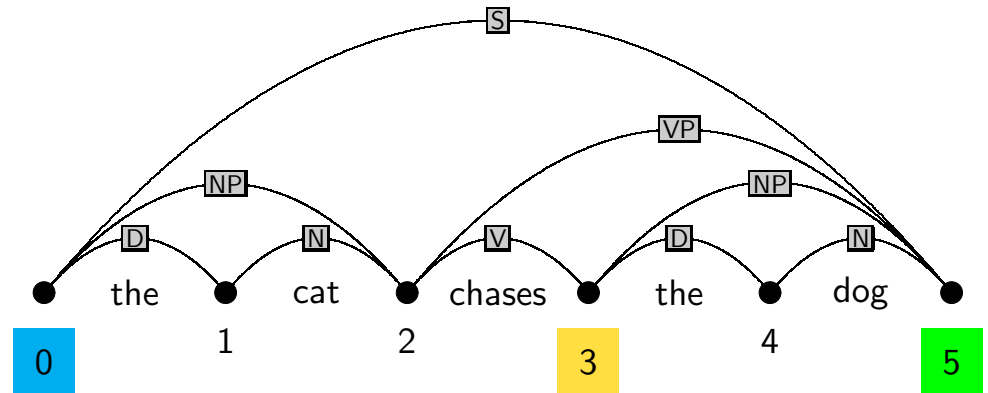


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

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

	1	2	3	4	5
0	d	np			s
1		n			
2			v		vp
3				d	np
4					n

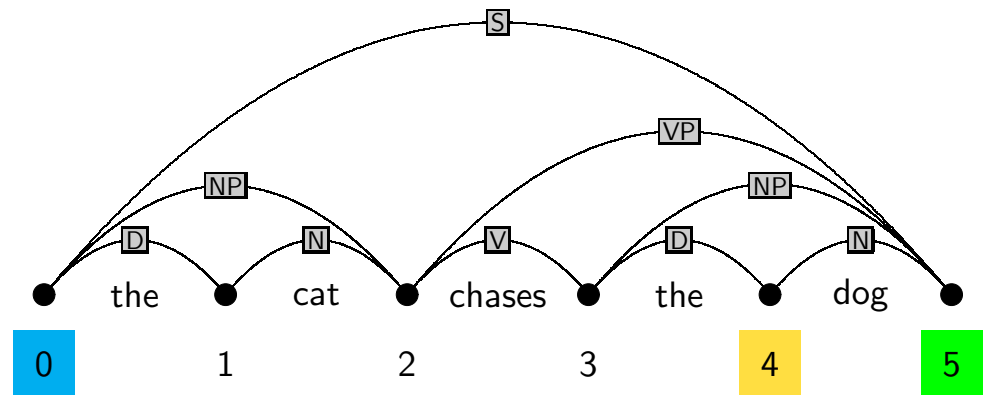


Example Application of the CYK Algorithm

$s \rightarrow np\ vp$ $d \rightarrow the$
 $np \rightarrow d\ n$ $n \rightarrow dog$
 $vp \rightarrow v\ np$ $n \rightarrow cat$
 $v \rightarrow chases$

$j = 5$
 $i = 0$
 $k = 4$

	1	2	3	4	5
0	d	np			s
1		n			
2			v		vp
3				d	np
4					n



Dynamic knowledge bases in PROLOG

- Declaration of a dynamic predicate: `dynamic/1` declaration, e.g:

```
:- dynamic chart/3.
```

to store facts of the form `chart(From,To,Category)`:

- Add a fact to the database: `assert/1`, e.g.:

```
assert(chart(1,3,np)).
```

Special versions `asserta/1/assertz/1` ensure adding facts first/last.

- Removing a fact from the database: `retract/1`, e.g.:

```
retract(chart(1,_,np)).
```

To remove all matching facts from the database use `retractall/1`

The CYK algorithm in PROLOG (parser/cky/cky.pl)

```
:- dynamic chart/3.                % chart(From,To,Category)
:- op(1100,xfx,'--->').          % Operator for grammar rules

% recognize(+WordList,?Startsymbol): top-level of CYK recognizer

recognize(String,Cat) :-
    retractall(chart(_,_,_)),      % initialize chart
    length(String,N),              % determine length of string
    fill_chart(String,0,N),        % call parser to fill the chart
    chart(0,N,Cat).                % check whether parse successful
```

```
% fill_chart(+WordList,+Current minus one,+Last)
```

```
% J-LOOP from 1 to n
```

```
fill_chart([],N,N).
```

```
fill_chart([W|Ws],JminOne,N) :-
```

```
    J is JminOne + 1,
```

```
    lexical_chart_fill(W,JminOne,J),
```

```
    %
```

```
    I is J - 2,
```

```
    syntactic_chart_fill(I,J),
```

```
    %
```

```
    fill_chart(Ws,J,N).
```

```
% lexical_chart_fill(+Word,+JminOne,+J)
% fill diagonal with preterminals

lexical_chart_fill(W,JminOne,J) :-
    (Cat ---> [W]),
    add_to_chart(JminOne,J,Cat),
    fail
; true.
```

```
% syntactic_chart_fill(+I,+J)
% I-LOOP from J-2 downto 0

syntactic_chart_fill(-1,_) :- !.
syntactic_chart_fill(I,J) :-
    K is I+1,
    build_phrases_from_to(I,K,J),
    %
    IminOne is I-1,
    syntactic_chart_fill(IminOne,J).
```

```
% build_phrases_from_to(+I,+Current-K,+J)
```

```
% K-LOOP from I+1 to J-1
```

```
build_phrases_from_to(_,J,J) :- !.
```

```
build_phrases_from_to(I,K,J) :-
```

```
    chart(I,K,B),
```

```
    chart(K,J,C),
```

```
    (A ---> [B,C]),
```

```
    add_to_chart(I,J,A),
```

```
    fail
```

```
; KplusOne is K+1,
```

```
    build_phrases_from_to(I,KplusOne,J).
```



```
% add_to_chart(+Cat,+From,+To): add if not yet there
add_to_chart(From,To,Cat) :-
    chart(From,To,Cat),
    !.
add_to_chart(From,To,Cat) :-
    assertz(chart(From,To,Cat)).
```