

Parsing

Cocke Younger Kasami (CYK)

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2021



Table of contents

- 1 Introduction
- 2 The recognizer
- 3 The CNF recognizer
- 4 CYK parsing
- 5 CYK with dotted productions
- 6 Bibliography

Introduction

The CYK parser is

- a **bottom-up** parser: we start with the terminals in the input string and subsequently compute recognized parse trees by going from already recognized rhs of productions to the non-terminal on the lefthand side.
- a **non-directional** parser: the checking for recognized components of a rhs in order to complete a lhs is not ordered; in particular, we cannot complete only parts of a rhs, everything in the rhs must be recognized in order to complete the lefthand category.

Independently proposed by Cocke, Kasami and Younger in the 60s.

Cocke and Schwartz (1970); Grune and Jacobs (2008); Hopcroft and Ullman (1979, 1994); Kasami (1965); Younger (1967)

The general recognizer (1)

We store the results in a $(n + 1) \times (n + 1)$ chart (table) C such that $A \in C_{i,l}$ iff $A \xRightarrow{*} w_i \dots w_{i+l-1}$.

In other words,

- i is the index of the first terminal in the relevant substring of w ; i ranges from 1 to $n + 1$ (the latter for an empty word following w_n)
- l is the length of the substring; l ranges from 0 to n .

All fields in the chart are initialized with \emptyset .

The general recognizer (2)

General CYK recognizer (for arbitrary CFGs):

```
 $C_{i,1} := \{w_i\}$  for all  $1 \leq i \leq n$  scan  
 $C_{i,0} := \{A \mid A \rightarrow \epsilon \in P\}$  for all  $i \in [1..n+1]$   $\epsilon$ -productions  
for all  $l \in [0..n]$ :  
  for all  $i_1 \in [1..n+1]$ :  
    repeat until chart does not change any more:  
      for every  $A \rightarrow A_1 \dots A_k$ :  
        if there are  $l_1, \dots, l_k \in [0..n]$  such that  
           $l_1 + \dots + l_k = l$  and  
           $A_j \in C_{i_j, l_j}$  with  $i_j = i_1 + l_1 \dots + l_{j-1}$ ,  
        then  $C_{i_1, l} := C_{i_1, l} \cup \{A\}$  complete
```

The general recognizer (3)

Example

$S \rightarrow ABC, A \rightarrow aA \mid \epsilon, B \rightarrow bB \mid \epsilon, C \rightarrow c.$

$w = aabbbc.$

l								
6	S							
5		S						
4			S					
3			B	S				
2	A		B	B	S			
1	a,A	a,A	b,B	b,B	b,B	c,C, S		
0	A,B	A,B	A,B	A,B	A,B	A,B	A,B	
	1	2	3	4	5	6	7	i

The general recognizer (4)

Parsing schema for CYK:

Items have three elements:

- $X \in N \cup T$: the nonterminal/terminal that spans a substring w_i, \dots, w_j of w ;
- the index i of the first terminal in the subsequence;
- the length $l = j - i$ of the subsequence.

Item form: $[X, i, l]$ with $X \in N \cup T$, $i \in [1..n + 1]$, $l \in [0..n]$.

The general recognizer (5)

Goal item: $[S, 1, n]$.

Deduction rules:

Scan: $\frac{}{[a, i, 1]} w_i = a$

ϵ -productions: $\frac{}{[A, i, 0]} A \rightarrow \epsilon \in P, i \in [1..n + 1]$

Complete: $\frac{[A_1, i_1, l_1], \dots, [A_k, i_k, l_k]}{[A, i_1, l]}$ $A \rightarrow A_1 \dots A_k \in P,$
 $l = l_1 + \dots + l_k,$
 $i_j = i_1 + l_1 \dots + l_{j-1}$

The general recognizer (6)

- Tabulation avoids problems with loops: nothing needs to be computed more than once.
- In each complete step, we have to check for all l_1, \dots, l_k . This is costly.
- Note, however, that we create a new chart entry (new item) only for combinations of already recognized parse trees. (No blind prediction as in Unger's parser.)
- With unary rules and ϵ -productions, an entry in field $C_{i,l}$ can be reused to compute a new entry in $C_{i,l}$. This is why the repeat until chart does not change any more loop is necessary.

The CNF recognizer (1)

A CFG is in Chomsky Normal Form iff all productions are either of the form $A \rightarrow a$ or $A \rightarrow BC$.

If the grammar has this form,

- we need to check only l_1, l_2 in a complete step, and
- we can be sure that to compute an entry in field $C_{i,l}$, we do not use another entry from field $C_{i,l}$. Consequently, we do not need the repeat until chart does not change any more loop.

The CNF recognizer (2)

The chart C is now an $n \times n$ -chart.

```
 $C_{i,1} := \{A \mid A \rightarrow w_i \in P\}$  scan  
for all  $l \in [1..n]$ :  
  for all  $i \in [1..n]$ :  
    for every  $A \rightarrow B \ C$ :  
      if there is a  $l_1 \in [1..l-1]$  such that  
         $B \in C_{i,l_1}$  and  $C \in C_{i+l_1,l-l_1}$ ,  
      then  $C_{i,l} := C_{i,l} \cup \{A\}$  complete
```

The CNF recognizer (3)

Parsing schema for CNF CYK:

Goal item: $[S, 1, n]$

Deduction rules:

Scan: $\frac{}{[A, i, 1]} A \rightarrow w_i \in P$

Complete: $\frac{[B, i, l_1], [C, i + l_1, l_2]}{[A, i, l_1 + l_2]} A \rightarrow B C \in P$

The CNF recognizer (4)

Example

$S \rightarrow C_a C_b \mid C_a S_B, S_B \rightarrow S C_b, C_a \rightarrow a, C_b \rightarrow b$. (From $S \rightarrow a S b \mid a b$ with transformation into CNF.)

$w = aaabbb$.

l							
6	S						
5		S_B					
4		S					
3			S_B				
2			S				
1	C_a	C_a	C_a	C_b	C_b	C_b	
	1	2	3	4	5	6	i
	a	a	a	b	b	b	

The CNF recognizer (5)

Time complexity: How many different instances of *scan* and *complete* are possible?

Scan: $\frac{}{[A, i, 1]} A \rightarrow w_i \in P$ $c_1 n$

Complete: $\frac{[B, i, l_1], [C, i + l_1, l_2]}{[A, i, l_1 + l_2]} A \rightarrow B C \in P$ $c_2 n^3$

Consequently, the time complexity of CYK for CNF is $\mathcal{O}(n^3)$.

The space complexity of CYK is $\mathcal{O}(n^2)$.

The CNF recognizer (6)

Control structures: there are two possible orders in which the chart can be filled:

- 1 *off-line order*: fill first row 1, then row 2 etc.:
for all $l \in [1..n]$: (length)
for all $i \in [1..n - l + 1]$: (start position)
compute $C_{i,l}$
- 2 *on-line order*: fill one diagonal after the other, starting with 1, 1 and proceeding from $k, 1$ to 1, k :
for all $k \in [1..n]$: (end position)
for all $l \in [1..k]$: (length)
compute $C_{k-l+1,l}$

The CNF recognizer (6)

- 1 Soundness of the algorithm: If $[A, i, l]$, then $A \xRightarrow{*} w_i \dots w_{i+l-1}$.
Proof via induction over deduction rules.
- 2 Completeness of the algorithm: If $A \xRightarrow{*} w_i \dots w_{i+l-1}$, then $[A, i, l]$.
Proof via induction over l .

CYK parsing (1)

We know that for every CFG G with $\epsilon \notin L(G)$ we can

- eliminate ϵ -productions,
- eliminate unary productions,
- eliminate useless symbols,
- transform into CNF,

and the resulting CFG G' is such that $L(G) = L(G')$.

Therefore, for every CFG, we can use the CNF recognizer after transformation.

How can we obtain a parser?

CYK parsing (2)

We need to do two things:

- turn the CNF recognizer into a parser, and
- if the original grammar was not in CNF, retrieve the original syntax from the CNF syntax.

CYK parsing (3)

To turn the CNF recognizer into a parser, we record not only non-terminal categories but whole productions with the positions and lengths of the rhs symbols in the chart (i.e., with backpointers):

```
 $C_{i,l} := \{A \rightarrow w_i \mid A \rightarrow w_i \in P\}$   
for all  $l \in [1..n]$ :  
  for all  $i \in [1..n]$ :  
    for every  $A \rightarrow B C$ :  
      if there is a  $l_1 \in [1..l-1]$  such that  
         $B \in C_{i,l_1}$  and  $C \in C_{i+l_1,l-l_1}$ ,  
      then  $C_{i,l} := C_{i,l} \cup \{A \rightarrow [B, i, l_1][C, i+l_1, l-l_1]\}$ 
```

We can then obtain a parse tree by traversing the productions from left to right, starting with every S -production in $C_{1,n}$.

CYK parsing (4)

Example

$S \rightarrow C_a C_b \mid C_a S_B, S_B \rightarrow S C_b, C_a \rightarrow a, C_b \rightarrow b, w = aaabbb$. (We write $A_{i,l}$ for $[A, i, l]$.)

$S \rightarrow$ $C_{a1,1} S_{B2,5}$					
	$S_B \rightarrow$ $S_{2,4} C_{b6,1}$				
	$S \rightarrow$ $C_{a2,1} S_{B3,3}$				
		$S_B \rightarrow$ $S_{3,2} C_{b5,1}$			
		$S \rightarrow$ $C_{a3,1} C_{b4,1}$			
$C_a \rightarrow a$	$C_a \rightarrow a$	$C_a \rightarrow a$	$C_b \rightarrow b$	$C_b \rightarrow b$	$C_b \rightarrow b$

CYK parsing (5)

From the CNF parse tree to the original parse tree:

First, we undo the CNF transformation:

- replace every $C_a \rightarrow a$ in the chart with a and replace every occurrence of C_a in a production with a .
- For all $l, i \in [1..n]$: If $A \rightarrow \alpha D_{i_D, l_D} \in C_{i,l}$ such that D is a new symbol introduced in the CNF transformation and $D \rightarrow \beta \in C_{i_D, l_D}$, then replace $A \rightarrow \alpha D_{i_D, l_D}$ with $A \rightarrow \alpha\beta$ in $C_{i,l}$.
- Finally remove all $D \rightarrow \gamma$ with D being a new symbol introduced in the CNF transformation from the chart.

CYK parsing (6)

Example

$S \rightarrow C_a C_b \mid C_a S_B, S_B \rightarrow S C_b, C_a \rightarrow a, C_b \rightarrow b, w = aaabbb$. New symbols: C_a, C_b, S_B . Elimination of C_a, C_b :

6	$S \rightarrow aS_{B2,5}$					
5		$S_B \rightarrow S_{2,4}b$				
4		$S \rightarrow aS_{B3,3}$				
3			$S_B \rightarrow S_{3,2}b$			
2			$S \rightarrow ab$			
1	a	a	a	b	b	b
	1	2	3	4	5	6

CYK parsing (7)

Example

$S \rightarrow C_a C_b \mid C_a S_B, S_B \rightarrow S C_b, C_a \rightarrow a, C_b \rightarrow b, w = aaabbb.$

Replacing of S_B in rhs:

6	$S \rightarrow aS_{2,4}b$					
5		$S_B \rightarrow S_{2,4}b$				
4		$S \rightarrow aS_{3,2}b$				
3			$S_B \rightarrow S_{3,2}b$			
2			$S \rightarrow ab$			
1	a	a	a	b	b	b
	1	2	3	4	5	6

CYK parsing (8)

Example

$S \rightarrow C_a C_b \mid C_a S_B, S_B \rightarrow S C_b, C_a \rightarrow a, C_b \rightarrow b, w = aaabbb.$

Elimination of S_B :

6	$S \rightarrow aS_{2,4}b$					
5						
4		$S \rightarrow aS_{3,2}b$				
3						
2			$S \rightarrow ab$			
1	a	a	a	b	b	b
	1	2	3	4	5	6

CYK parsing (9)

Undo the elimination of unary productions:

- For every $A \rightarrow \beta$ in $C_{i,l}$ that has been added in removing of the unary productions to replace $B \rightarrow \beta'$ (β' is β without chart indices): replace A with B in this entry in $C_{i,l}$.
- For every unary production $A \rightarrow B$ in the original grammar and for every $B \rightarrow \beta \in C_{i,l}$: add $A \rightarrow B_{i,l}$ to $C_{i,l}$. Repeat this until chart does not change any more.

CYK parsing (10)

Undo the elimination of ϵ -productions:

- Add a row with $l = 0$ and a column with $i = n + 1$ to the chart.
- Fill row 0 as in the general case using the original CFG grammar (tabulating productions).
- For every $A \rightarrow \beta$ in $C_{i,l}$ that has been added in removing the ϵ -productions: add the deleted nonterminals to β with the position of the preceding non-terminal as starting position (or i if it is the first in the rhs) and with length 0.

CYK parsing (11)

Terminal Filter: Observation: Information on the obligatory presence of terminals might get lost in the CNF transformation:

$S \rightarrow aSb$ (requires an a , an S and a b) \rightsquigarrow $S \rightarrow C_a S_B$ (requires an a and an S and a b) and $S_B \rightarrow S C_b$ (requires an S and a b)

Consider an input $babb$:

- In a CYK parser with the original grammar, we would derive $[S, 2, 2]$ and $[b, 4, 1]$ but we could not apply $S \rightarrow aSb$.
- In the CNF grammar, we would have $[S, 2, 2]$ and $[C_b, 4, 1]$ and then we could apply $S_B \rightarrow S C_b$ and obtain $[S_B, 2, 3]$ even though the only way to continue with S_B in a rhs is with $S \rightarrow C_a S_B$ which is not possible since the first terminal is not an a .

CYK parsing (12)

Solution: add an additional check:

- Every new non-terminal D introduced for binarization stands (deterministically) for some substring β of a rhs $\alpha\beta$.
Ex: S_B in our sample grammar stands for Sb .
- Every terminal in this rhs to the left of β , i.e., every terminal in α must necessarily be present to the left for a deduction of a D that leads to a goal item.
Ex: S_B can only lead to a goal item if to its left we have an a .
- **Terminal Filter:** During CNF transformation, for every non-terminal D introduced for binarization, record the sets of terminals in the rhs to the left of the part covered by D .
During parsing, check for the presence of these terminals to the left of the deduced D item.

CYK with dotted productions (1)

CNF leads to a binarization: In each completion, only two items are combined.

Such a binarization can be obtained by using dotted productions:

- We process the rhs of a production from left to right.
- In each complete step, a production $A \rightarrow \alpha \bullet X\beta$ is combined with an X whose span starts at the end of the α -span. The result is a production $A \rightarrow \alpha X \bullet \beta$.
- We start with the completed terminals and their spans.

Note that this version of CYK is directional.

CYK with dotted productions (2)

Parsing schema for the general version (allowing also for ε -productions):

Goal items: $[S \rightarrow \alpha \bullet, 0, n]$ for all S -productions $S \rightarrow \alpha$.

Deduction rules:

Predict (axioms): $\frac{}{[A \rightarrow \bullet \alpha, i, i]} A \rightarrow \alpha \in P, i \in [0..n]$

Scan: $\frac{[A \rightarrow \alpha \bullet a \beta, i, j]}{[A \rightarrow \alpha a \bullet \beta, i, j + 1]} w_{j+1} = a$

Complete: $\frac{[A \rightarrow \alpha \bullet B \beta, i, j][B \rightarrow \gamma \bullet, j, k]}{[A \rightarrow \alpha B \bullet \beta, i, k]}$

CYK with dotted productions (3)

Parsing schema including passive items (just a non-terminal or terminal, no dotted production) and assuming an ε -free CFG:

Goal item: $[S, 0, n]$

Deduction rules:

Scan (axioms): $\frac{}{[a, i, i + 1]} w_{i+1} = a$

Left-corner predict: $\frac{[X, i, j]}{[A \rightarrow X \bullet \alpha, i, j]} A \rightarrow X\alpha \in P, X \in N \cup T$

Complete: $\frac{[A \rightarrow \alpha \bullet X\beta, i, j][X, j, k]}{[A \rightarrow \alpha X \bullet \beta, i, k]}$

Publish: $\frac{[A \rightarrow \alpha \bullet, i, j]}{[A, i, j]}$

(This is actually a deduction-based version of left-corner parsing.)

CYK with dotted productions (4)

Example

Example (without ϵ -productions, left-corner parsing): $S \rightarrow ab \mid aSb$

$w = aabb$

4	$S \rightarrow aSb\bullet$ S			b
3	$S \rightarrow aS\bullet b$	$S \rightarrow ab\bullet$ S	b	
2		a $S \rightarrow a\bullet b$ $S \rightarrow a\bullet Sb$		
1	a $S \rightarrow a\bullet b$ $S \rightarrow a\bullet Sb$			
	0	1	2	3

CYK with dotted productions (5)

What about time complexity?

The most complex operation, *complete*, involves only three indices i, j, k ranging from 0 to n :

$$\text{Complete: } \frac{[A \rightarrow \alpha \bullet X\beta, i, j][X, j, k]}{[A \rightarrow \alpha X \bullet \beta, i, k]}$$

Consequently, the time complexity is $\mathcal{O}(n^3)$, as in the CNF case.

But: the data structure required for representing a parse item with a dotted production is slightly more complex than what is needed for simple passive items.

Conclusion

- CYK is a non-directional bottom-up parser.
- If used with CNF, it is very efficient. Time complexity is $\mathcal{O}(n^3)$.
- The transformation into CNF can be undone after parsing, i.e., we still have a parser for arbitrary CFGs (as long as ϵ is not in the language).
- Instead of explicitly binarizing, we can use dotted productions and move through the righthand sides of productions step by step from left to right, which also leads to $\mathcal{O}(n^3)$.

Bibliography

- Cocke, J. and Schwartz, J. T. (1970). Programming languages and their compilers: Preliminary notes. Technical report, CIMS, NYU. 2nd revised edition.
- Grune, D. and Jacobs, C. (2008). *Parsing Techniques. A Practical Guide*. Monographs in Computer Science. Springer. Second Edition.
- Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison Wesley.
- Hopcroft, J. E. and Ullman, J. D. (1994). *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison Wesley, 3. edition.
- Kasami, T. (1965). An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, AFCRL.
- Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3 . *Inform. Control*, 10(2):189–20.