The Cocke-Young-Kasami (CYK) Algorithm

The problem

Given a word w and a grammar G, can w be generated by G? (Membership Problem)

Regular languages case: we can check it in O(|w|) time (just run the equivalent F.A.)

Context-free grammars: the problem is that we do not know in advance how a word is split in general by rules line A—>BC.
How many symbols of w come from B and how many from C?

The problem looks combinatorial but actually thanks to a dynamic programming technique (efficient re-use of previous sub-problem results) we can devise a $O(|w|^3)$ algorithm: the CYK algorithm,

Idea

1) Consider all the substrings of w

2) Compute which non-terminals can generate each substring

3) Work your way bottom-up, starting from short substrings to longer ones.

⟶ Dynamic Programming

Example

Consider the word w = baaba

Can the grammar G:

$$S \rightarrow AB \mid BC$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow CC \mid b$$

$$C \rightarrow AB \mid a$$

generate it?

STEP 1: Build the table of substrings

|w| = 5	T_{15} baaba	-	-	-	-
|w| = 4	T_{14} baab	T_{25} aaba	-	-	-
|w| = 3	T_{13} baa	T_{24} aab	T_{35} aba	-	-
|w| = 2	T_{12} ba	T_{23} aa	T_{34} ab	T_{45} ba	-
Base-case |w| = 1	T_{11} b	T_{22} a	T_{33} a	T_{44} b	T_{55} a

w = baaba

Notation: $A \in T_{ij}$ = if the non-terminal A generates the string $w_i \ldots w_j$
T are therefore **sets** of non-terminals to be determined.

STEP 2: Determine T for the base-case

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	-	-	-	-
		-	-	-
			-	-
				-
{B}	{A,C}	{A,C}	{B}	{A,C}

w = baaba

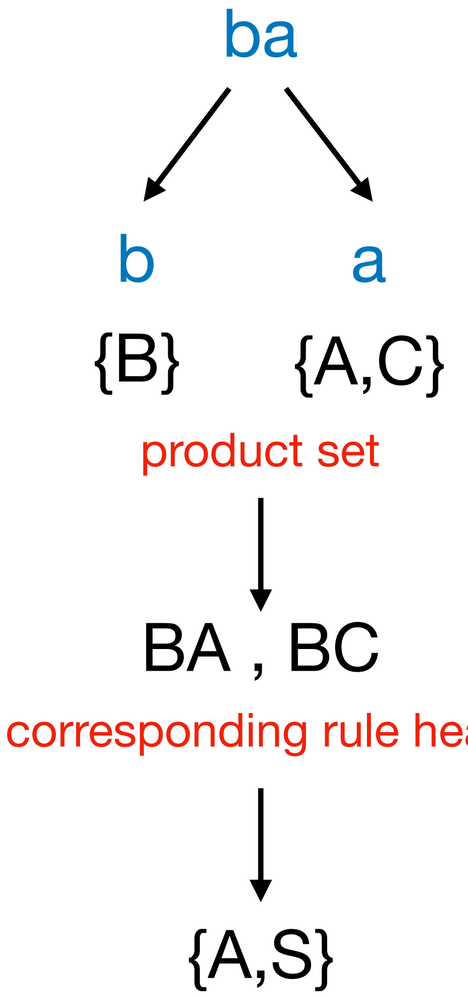
STEP 3: Determine the first row

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$



	-	-	-	-
		-	-	-
			-	-
T_{12}				-
{B}	{A,C}	{A,C}	{B}	{A,C}

w = baaba

STEP 3: Determine the first row

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	-	-	-	-
		-	-	-
			-	-
{A,S}				-
{B}	{A,C}	{A,C}	{B}	{A,C}

w = baaba

STEP 4: Finish the first row

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	-	-	-	-
		-	-	-
			-	-
{A,S}	{B}	{S,C}	{S,A}	-
{B}	{A,C}	{A,C}	{B}	{A,C}

w = baaba

STEP 5: Generic element, example with T_{24}

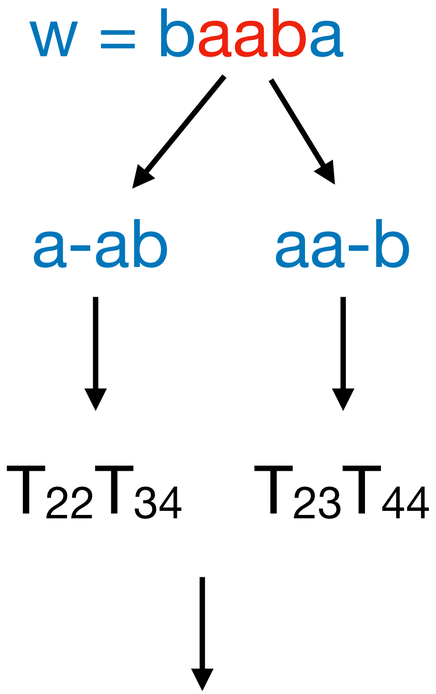
$S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

w = baaba



	-	-	-	-
		-	-	-
	T_{24}		-	-
{A,S}	{B}	{S,C}	{S,A}	-
{B}	{A,C}	{A,C}	{B}	{A,C}

$$T_{22}T_{34} \cup T_{23}T_{44} = \{A, C\}\{S, C\} \cup \{B\}\{B\} = \{AS, AC, CS, CC, BB\}$$

Only this is a rule-body. The head is B

STEP 6: Finish the table

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

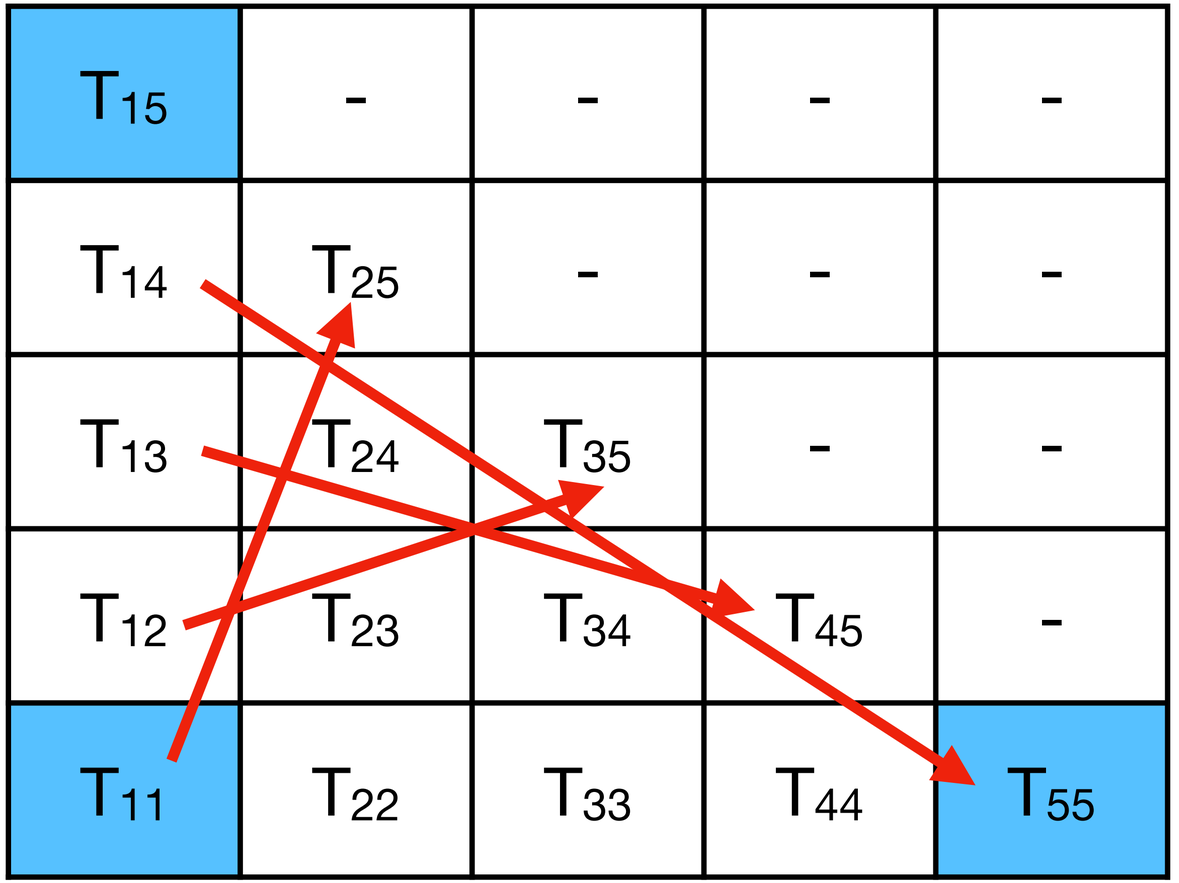
$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

Last element to calculate.
It contains the start rule S.
WORD ACCEPTED!

{S,A,C}	-	-	-	-
-	{S,A,C}	-	-	-
-	{B}	{B}	-	-
{A,S}	{B}	{S,C}	{S,A}	-
{B}	{A,C}	{A,C}	{B}	{A,C}

String breaking pattern



We have to compare at most n pairs of previously computed sets: $(T_{ii}, T_{i+1,j})$, $(T_{i,i+1}, T_{i+2,j})$…$(T_{i,j-1}, T_{jj})$

Conclusion

The CYK algorithm tests if a word can be generated by a CF grammar.

If S belongs to the last table entry (top-left), the word is accepted.

It is a dynamic-programming algorithm: previous results are used for producing the new one.

Given $n=|w|$, the algorithm's time scales as $O(n^3)$:

- Loop over the string length
- Loop over starting positions
- Loop over possible splittings

Memory-wise, the table contains $n(n-1)/2$ entries, so space scales as $O(n^2)$