CSCI 742 - Compiler Construction

Lecture 18

Cocke-Younger-Kasami (CYK) Algorithm

Instructor: Hossein Hojjat

February 26, 2018

Recap: Chomsky Normal Form (CNF)

A CFG is in Chomsky Normal Form if each rule is of the form

$$A \to BC$$
$$A \to a$$

where

- a is any terminal
- A,B,C are non-terminals
- B, C cannot be start variable

We allow the rule $S \to \epsilon$ if $\epsilon \in L$

Recap: Conversion to CNF

1. remove unproductive non-terminals (optional)
2. remove unreachable non-terminals (optional)
3. make terminals occur alone on right-hand side
4. reduce arity of every production to ≤ 2
5. remove ϵ-production rules $X \to \epsilon$
6. remove unit productions $X \to Y$
7. unproductive non-terminals
8. unreachable non-terminals

CYK (Cocke-Younger-Kasami)

- One of the earliest recognition and parsing algorithms
- Bottom-up parsing (starts with words)
- Independently developed by J. Cocke, D. Younger, T. Kasami (CYK)
- Build solutions compositionally from sub-solutions
- Based on a "dynamic programming" approach

CYK algorithm: Basic idea

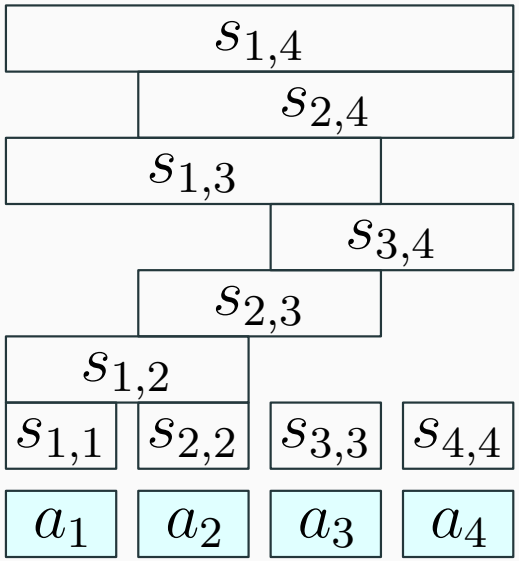
For a grammar G and a word w:

- For every substring v_1 of length 1, find all non-terminals A such that $A \Rightarrow^* v_1$
- For every substring v_2 of length 2, find all non-terminals A such that $A \Rightarrow^* v_2$

  $\vdots$
- For the unique substring w of length $|w|$, find all non-terminals A such that $A \Rightarrow^* w$

Check whether S belongs to the last set

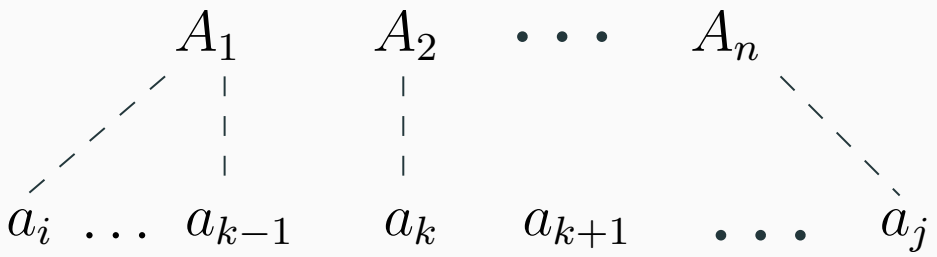
Substring Recognition



- word: $w = a_1 a_2 \cdots a_n$
- $X_{i,j}$ set of all non-terminals deriving the substring $s_{i,j}$
- Start non-terminal S derives w $(= s_{1,n})$ if and only if S is a member of $X_{1,n}$

Substring Recognition

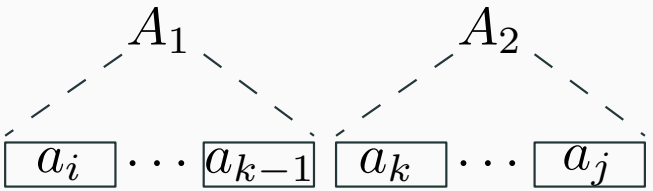
- How to check if the RHS of rule $A \rightarrow A_1 A_2 \cdots A_n$ derives $s_{i,j}$?



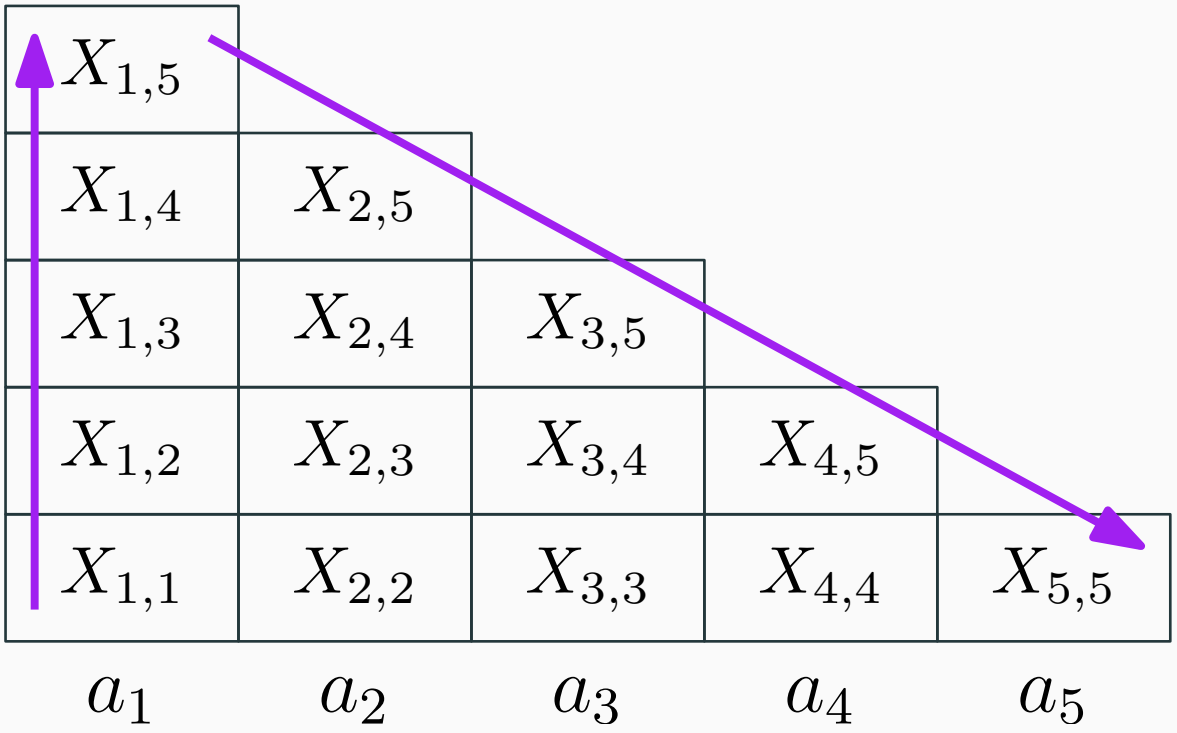
(i < k < j)

- We need to check all the possible matching of substrings with A_i non-terminals
- Results of checking subproblem solutions are reusable
- Subproblem results are computed once and then memoized for later

Matching in CNF:



Recognition Table



Each row corresponds to one length of substrings

- Bottom Row: words of length 1
- Second from Bottom Row: words of length 2
- $\vdots$
- Top Row: word w

Recognition Table

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
a_1	a_2	a_3	a_4	a_5

if B belongs to $X_{1,1}$ and C to $X_{2,5}$ and if $A \to BC$ is a grammar rule then add A to $X_{1,5}$

Each row corresponds to one length of substrings

- Bottom Row: words of length 1
- Second from Bottom Row: words of length 2
- $\vdots$
- Top Row: word w

Recognition Table

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
a_1	a_2	a_3	a_4	a_5

Each row corresponds to one length of substrings

- Bottom Row: words of length 1
- Second from Bottom Row: words of length 2
- $\vdots$
- Top Row: word w

Recognition Table

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
a_1	a_2	a_3	a_4	a_5

Each row corresponds to one length of substrings

- Bottom Row: words of length 1
- Second from Bottom Row: words of length 2
- $\vdots$
- Top Row: word w

Recognition Table

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
a_1	a_2	a_3	a_4	a_5

Each row corresponds to one length of substrings

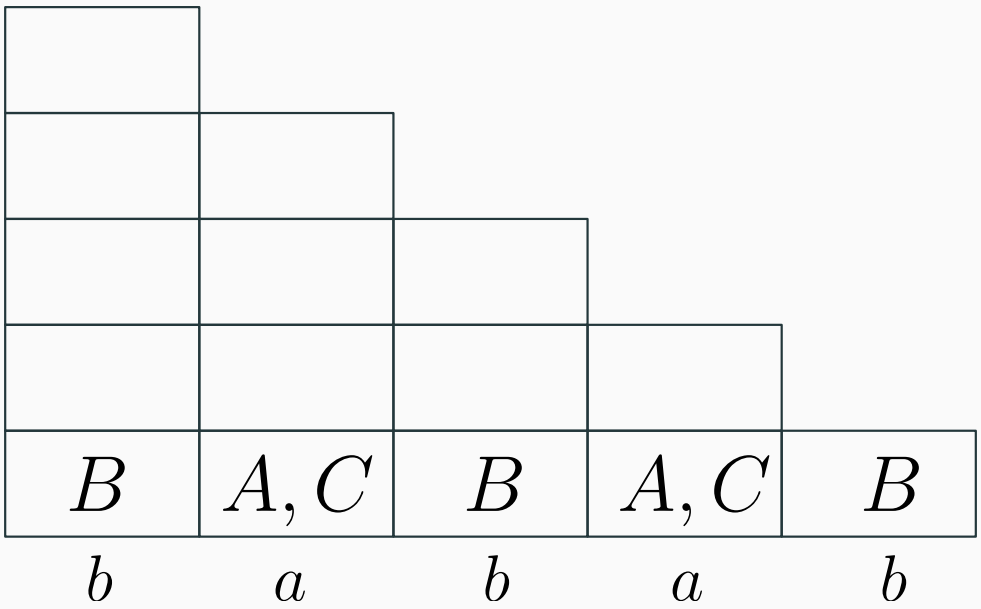
- Bottom Row: words of length 1
- Second from Bottom Row: words of length 2
- $\vdots$
- Top Row: word w

Example

grammar:

$$S \to AB \mid CB$$
$$A \to BA \mid a$$
$$B \to BC \mid b$$
$$C \to AC \mid a$$

intput: $babab$

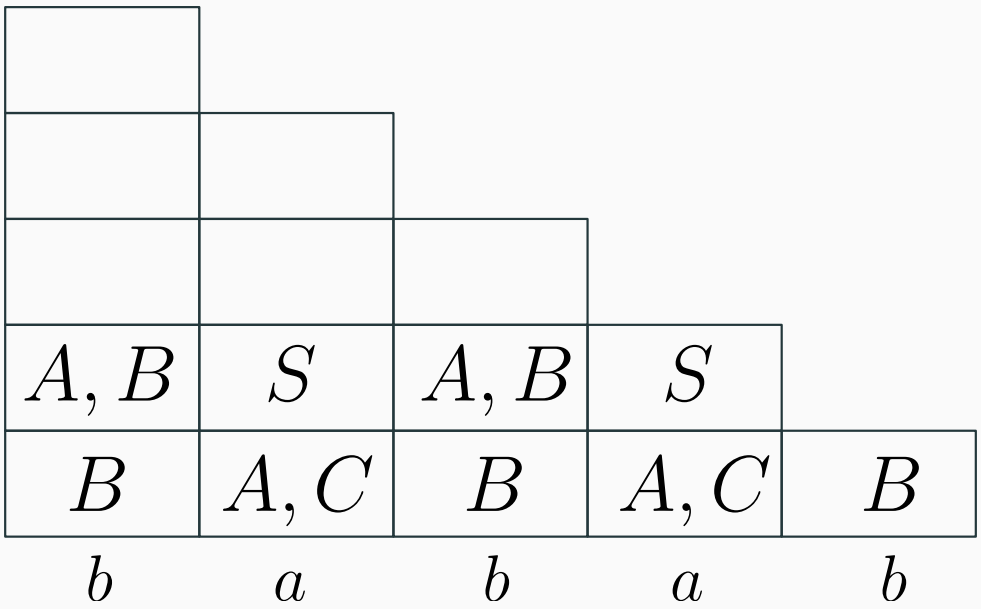


Example

grammar:

$$S \to AB \mid CB$$
$$A \to BA \mid a$$
$$B \to BC \mid b$$
$$C \to AC \mid a$$

intput: $babab$



Example

grammar:

$$S \to AB \mid CB$$
$$A \to BA \mid a$$
$$B \to BC \mid b$$
$$C \to AC \mid a$$

intput: $babab$

S	S	S		
A, B	S	A, B	S	
B	A, C	B	A, C	B
b	a	b	a	b

Example

grammar:

$$S \to AB \mid CB$$
$$A \to BA \mid a$$
$$B \to BC \mid b$$
$$C \to AC \mid a$$

intput: $babab$

S, A				
S	S	S		
A, B	S	A, B	S	
B	A, C	B	A, C	B
b	a	b	a	b

Example

grammar:

$$S \to AB \mid CB$$
$$A \to BA \mid a$$
$$B \to BC \mid b$$
$$C \to AC \mid a$$

intput: $babab$

S				
S, A				
S	S	S		
A, B	S	A, B	S	
B	A, C	B	A, C	B
b	a	b	a	b

CYK algorithm

- Standard CYK operates only on context-free grammars in Chomsky Normal Form (CNF)
- To illustrate some of the benefits of CNF we use a non-CNF grammar

Example word: `12.3e+4`

$$\begin{aligned}
\text{Number} &\to \text{Integer} \mid \text{Real} \\
\text{Integer} &\to \text{Digit} \mid \text{Integer Digit} \\
\text{Real} &\to \text{Integer Fraction Scale} \\
\text{Fraction} &\to . \ \text{Integer} \\
\text{Scale} &\to \mathrm{e} \ \text{Sign Integer} \mid \epsilon \\
\text{Digit} &\to 0 \mid 1 \mid \cdots \mid 8 \mid 9 \\
\text{Sign} &\to + \mid -
\end{aligned}$$

Example word: `12.3e+4`

Number $\rightarrow$ Integer | Real

Integer $\rightarrow$ Digit | Integer Digit

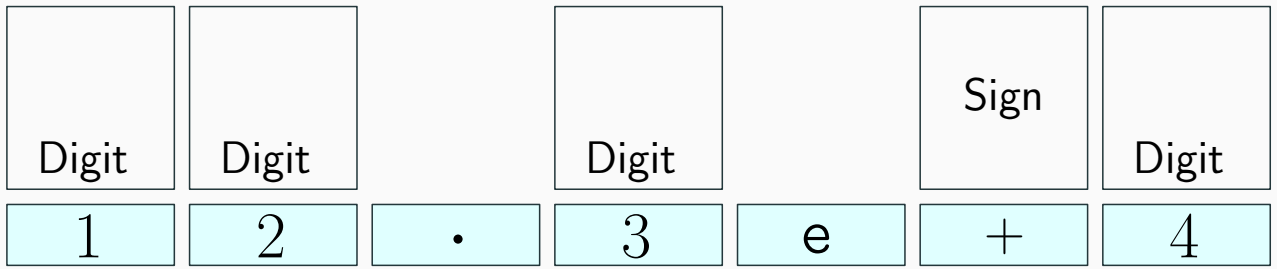
Real $\rightarrow$ Integer Fraction Scale

Fraction $\rightarrow$. Integer

Scale $\rightarrow$ e Sign Integer | ϵ

Digit $\rightarrow 0 \mid 1 \mid \cdots \mid 8 \mid 9$

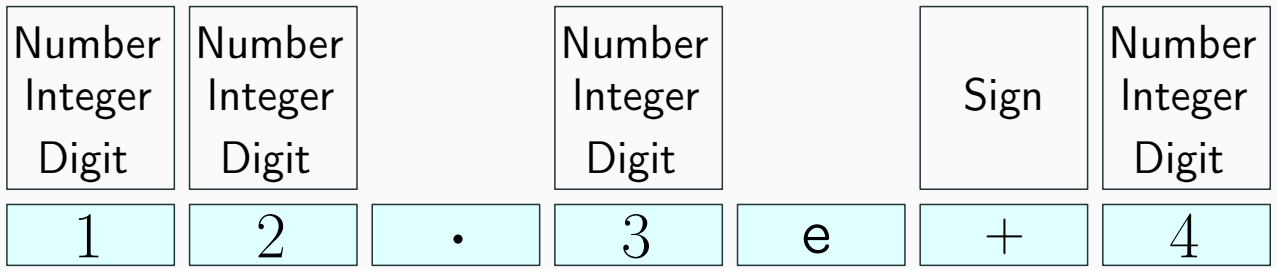
Sign $\rightarrow + \mid -$



- We start with substrings of length 1

Example word: `12.3e+4`

$$
\begin{aligned}
\text{Number} &\to \text{Integer} \mid \text{Real} \\
\text{Integer} &\to \text{Digit} \mid \text{Integer Digit} \\
\text{Real} &\to \text{Integer Fraction Scale} \\
\text{Fraction} &\to . \ \text{Integer} \\
\text{Scale} &\to \text{e Sign Integer} \mid \epsilon \\
\text{Digit} &\to 0 \mid 1 \mid \cdots \mid 8 \mid 9 \\
\text{Sign} &\to + \mid -
\end{aligned}
$$



- We start with substrings of length 1
- Grammar has unit-rules: after finding a non-terminal, we need to repetitively search for other non-terminals that derive the same word
- Chain of rules: Number $\Rightarrow$ Integer $\Rightarrow$ Digit

Example word: `12.3e+4`

Number → Integer | Real

Integer → Digit | Integer Digit

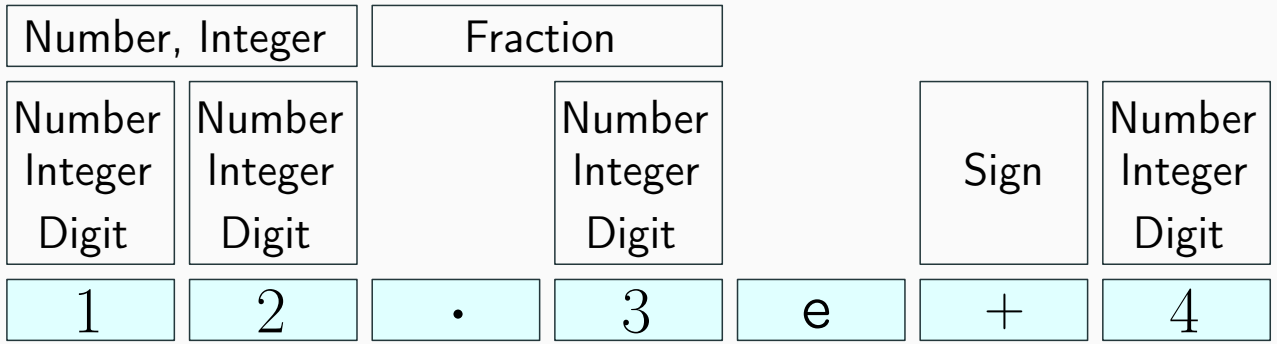
Real → Integer Fraction Scale

Fraction → . Integer

Scale → e Sign Integer | ϵ

Digit → 0 | 1 | ⋯ | 8 | 9

Sign → + | −



Example word: `12.3e+4`

Number → Integer | Real

Integer → Digit | Integer Digit

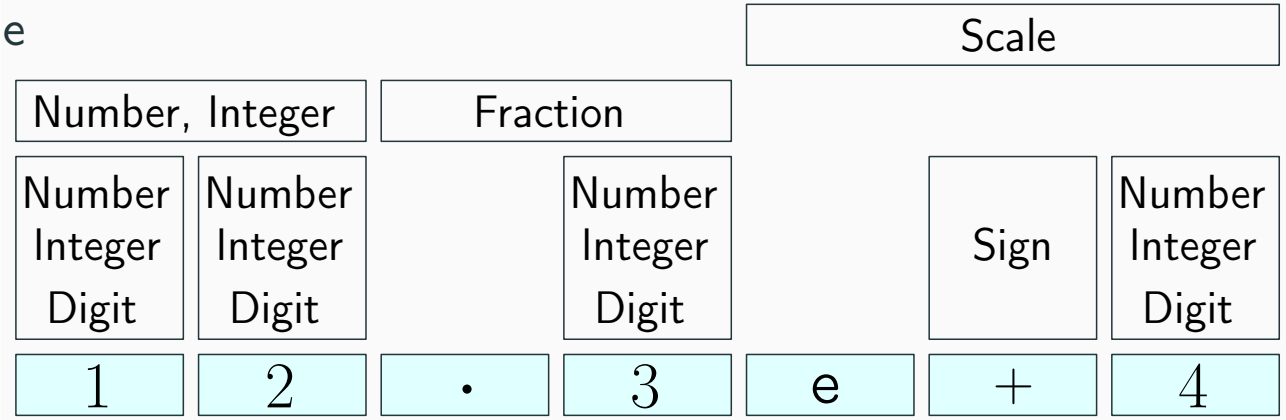
Real → Integer Fraction Scale

Fraction → . Integer

Scale → e Sign Integer | ϵ

Digit → 0 | 1 | ⋯ | 8 | 9

Sign → + | −



Example word: `12.3e+4`

Number → Integer | Real

Integer → Digit | Integer Digit

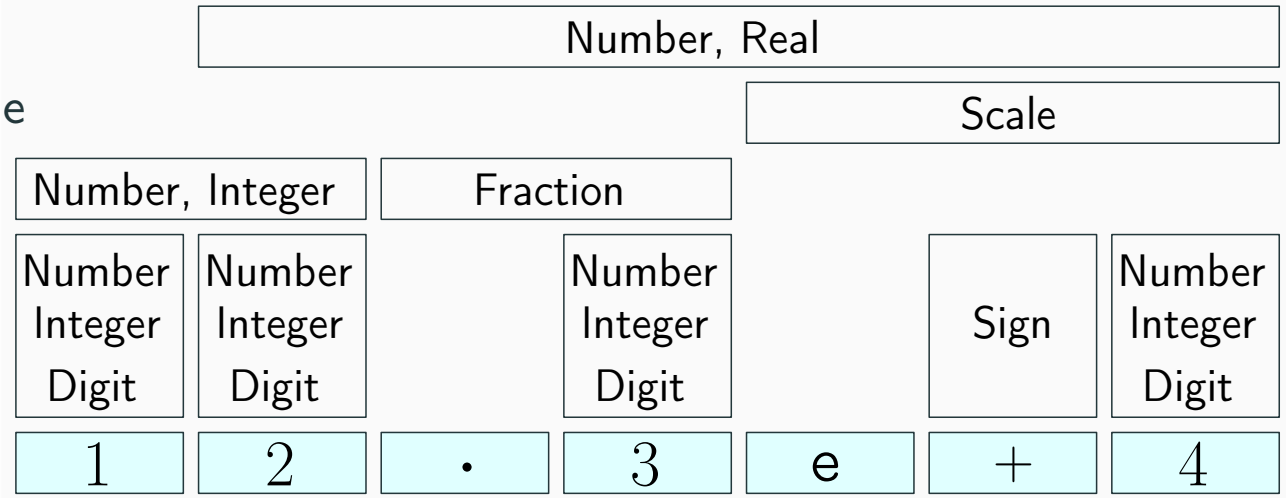
Real → Integer Fraction Scale

Fraction → . Integer

Scale → e Sign Integer | ϵ

Digit → 0 | 1 | ⋯ | 8 | 9

Sign → + | −



Example word: `12.3e+4`

Number → Integer | Real

Integer → Digit | Integer Digit

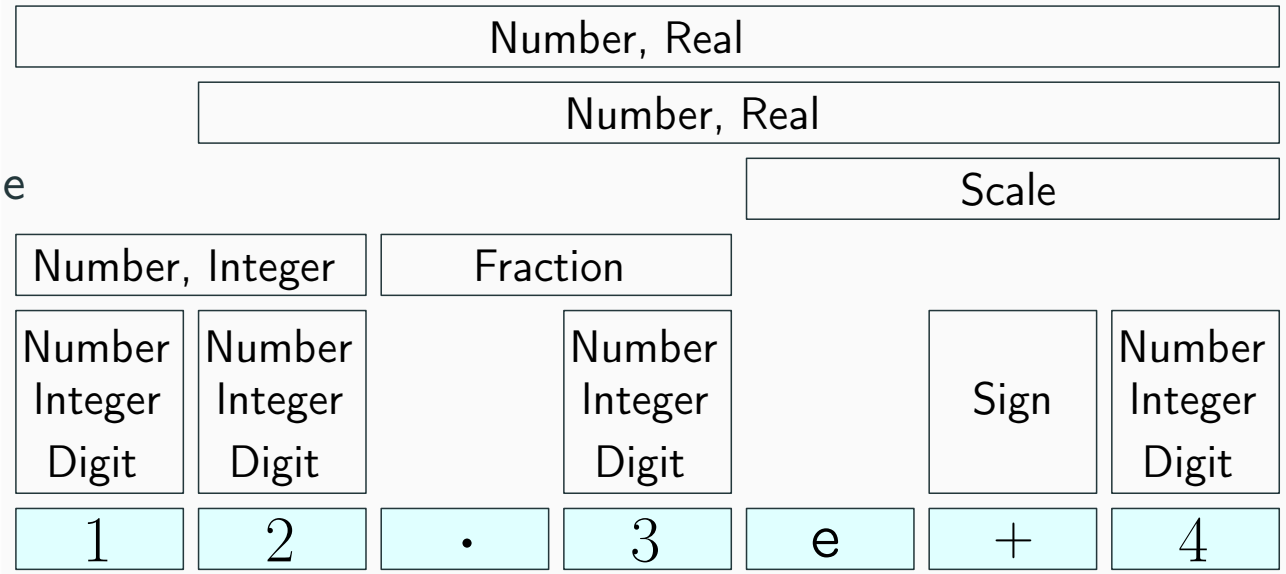
Real → Integer Fraction Scale

Fraction → . Integer

Scale → e Sign Integer | ϵ

Digit → 0 | 1 | ⋯ | 8 | 9

Sign → + | −



Example word: `12.3`

Number → Integer | Real

Integer → Digit | Integer Digit

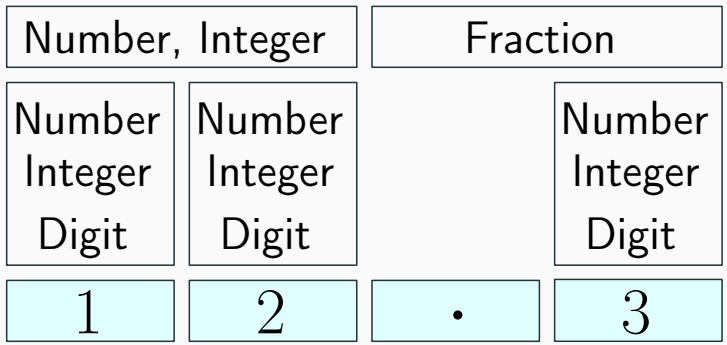
Real → Integer Fraction Scale

Fraction → . Integer

Scale → e Sign Integer | ϵ

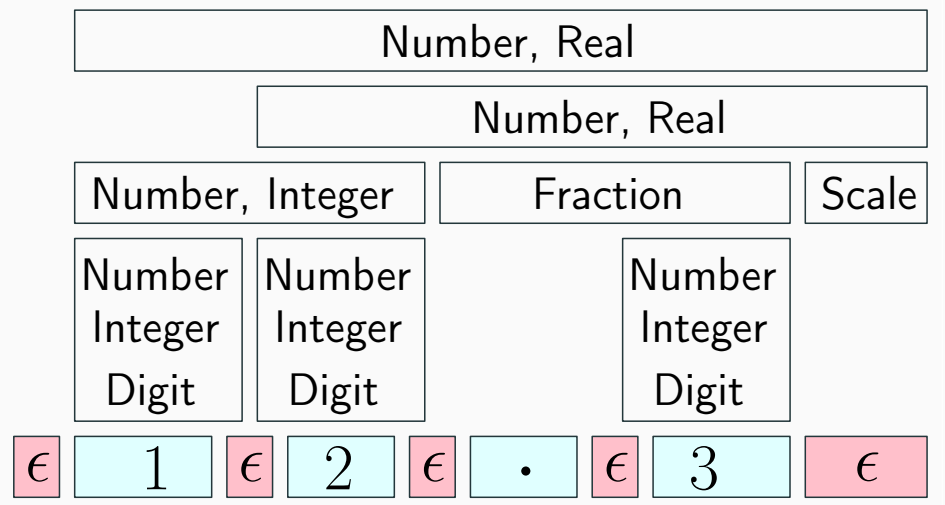
Digit → 0 | 1 | ⋯ | 8 | 9

Sign → + | −



Example word: 12.3

$$
\begin{aligned}
\text{Number} &\to \text{Integer} \mid \text{Real} \\
\text{Integer} &\to \text{Digit} \mid \text{Integer Digit} \\
\text{Real} &\to \text{Integer Fraction Scale} \\
\text{Fraction} &\to . \ \text{Integer} \\
\text{Scale} &\to e \ \text{Sign Integer} \mid \epsilon \\
\text{Digit} &\to 0 \mid 1 \mid \cdots \mid 8 \mid 9 \\
\text{Sign} &\to + \mid -
\end{aligned}
$$



- We neglected ϵ-production rules until now
- ϵ-rules can be used between any two symbols, at the beginning and at the end

Exercise

Question:

- What is the time and space complexity of CYK?
  (Assume size of input word is n)

Exercise

Question:

- What is the time and space complexity of CYK?
  (Assume size of input word is n)

Answer:

Space complexity:

- $n \times n$ table with lists up to size of non-terminals: $O(n^2)$

Time complexity:

- if $|G|$ is size of grammar $O(n^3|G|)$

Exercise

Question:

How can you change the CYK algorithm to count the number of parse trees of a given input string?

Exercise

Question:

How can you change the CYK algorithm to count the number of parse trees of a given input string?

Answer:

- Number of parse trees for $s_{i,j}$ is the sum of the number parse trees in each partition of $s_{i,j}$ into two
- Given a partition $s_{i,p}$ and $s_{p+1,j}$ of $s_{i,j}$, the number of parse trees for the partition is the product of the number of parse trees for $s_{i,p}$ and $s_{p+1,j}$

Exercise

Show the CYK Algorithm with the following example:

- CNF grammar:

$$S \to AB \mid BC$$
$$A \to BA \mid a$$
$$B \to CC \mid b$$
$$C \to AB \mid a$$

- Input: $ababa$

CYK Problem

- Parsers usually parse the original grammar without further modification
- Grammar reflects the actual structure of language
- Conversion to CNF can make it difficult to keep the intended structure