



CSCI 742 - Compiler Construction

Lecture 18

Cocke-Younger-Kasami (CYK) Algorithm

Instructor: Hossein Hojjat

February 26, 2017

Recap: Chomsky Normal Form (CNF)

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

$$A \rightarrow BC$$

$$A \rightarrow a$$

where

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

We allow the rule $S \rightarrow \epsilon$ 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 $X \rightarrow \epsilon$
6. remove unit productions $X \rightarrow Y$
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

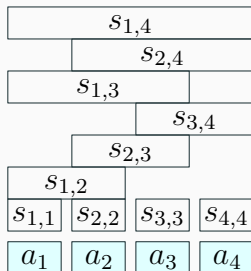
CYK algorithm: Basic idea

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

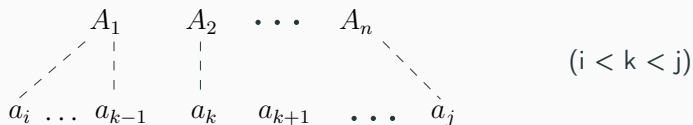
Substring Recognition



- word: $w = a_1a_2 \cdots a_n$
- $X_{i,j}$ 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 $X_{1,n}$

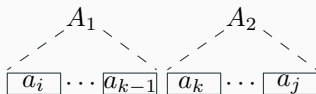
Substring Recognition

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



- 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

| | | | | |
|-----------|-----------|-----------|-----------|-----------|
| $X_{1,5}$ | | | | |
| $X_{1,4}$ | $X_{2,5}$ | | | |
| $X_{1,3}$ | $X_{2,4}$ | $X_{3,5}$ | | |
| $X_{1,2}$ | $X_{2,3}$ | $X_{3,4}$ | $X_{4,5}$ | |
| $X_{1,1}$ | $X_{2,2}$ | $X_{3,3}$ | $X_{4,4}$ | $X_{5,5}$ |
| 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

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:

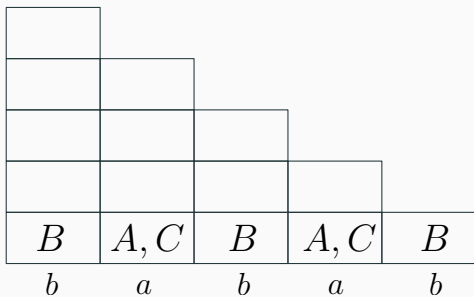
$$S \rightarrow AB \mid CB$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow BC \mid b$$

$$C \rightarrow AC \mid a$$

input: *babab*



Example

grammar:

$$S \rightarrow AB \mid CB$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow BC \mid b$$

$$C \rightarrow AC \mid a$$

input: *babab*

| | | | | |
|-------------|-------------|-------------|-------------|----------|
| | | | | |
| | | | | |
| | | | | |
| <i>A, B</i> | <i>S</i> | <i>A, B</i> | <i>S</i> | |
| <i>B</i> | <i>A, C</i> | <i>B</i> | <i>A, C</i> | <i>B</i> |
| <i>b</i> | <i>a</i> | <i>b</i> | <i>a</i> | <i>b</i> |

Example

grammar:

$$S \rightarrow AB \mid CB$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow BC \mid b$$

$$C \rightarrow AC \mid a$$

input: *babab*

| | | | | |
|-------------|-------------|-------------|-------------|----------|
| | | | | |
| | | | | |
| <i>S</i> | <i>S</i> | <i>S</i> | | |
| <i>A, B</i> | <i>S</i> | <i>A, B</i> | <i>S</i> | |
| <i>B</i> | <i>A, C</i> | <i>B</i> | <i>A, C</i> | <i>B</i> |
| <i>b</i> | <i>a</i> | <i>b</i> | <i>a</i> | <i>b</i> |

Example

grammar:

$$S \rightarrow AB \mid CB$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow BC \mid b$$

$$C \rightarrow AC \mid a$$

input: *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:

$$S \rightarrow AB \mid CB$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow BC \mid b$$

$$C \rightarrow AC \mid a$$

input: *babab*

| | | | | | |
|-------------|-------------|-------------|-------------|----------|--|
| <i>S</i> | | | | | |
| <i>S, A</i> | | | | | |
| <i>S</i> | <i>S</i> | <i>S</i> | | | |
| <i>A, B</i> | <i>S</i> | <i>A, B</i> | <i>S</i> | | |
| <i>B</i> | <i>A, C</i> | <i>B</i> | <i>A, C</i> | <i>B</i> | |
| <i>b</i> | <i>a</i> | <i>b</i> | <i>a</i> | <i>b</i> | |

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

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 | 1 | \dots | 8 | 9

Sign \rightarrow + | -

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | . | 3 | e | + | 4 |
|---|---|---|---|---|---|---|

Example word: 12.3e+4

Number \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

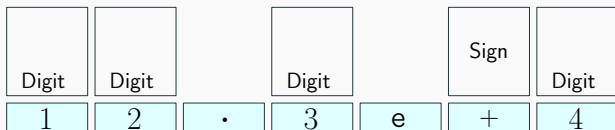
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow e Sign Integer | ϵ

Digit \rightarrow 0 | 1 | \dots | 8 | 9

Sign \rightarrow + | -



- We start with substrings of length 1

Example word: 12.3e+4

Number \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

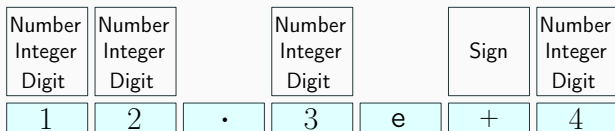
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow e Sign Integer | ϵ

Digit \rightarrow 0 | 1 | \dots | 8 | 9

Sign \rightarrow + | -



- 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 \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

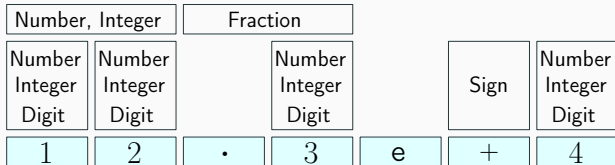
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow e Sign Integer | ϵ

Digit \rightarrow 0 | 1 | ... | 8 | 9

Sign \rightarrow + | -



Example word: 12.3e+4

Number \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

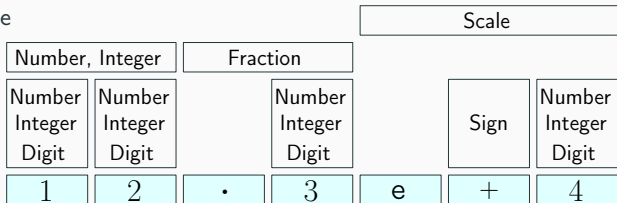
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow e Sign Integer | ϵ

Digit \rightarrow 0 | 1 | ... | 8 | 9

Sign \rightarrow + | -



Example word: 12.3e+4

Number \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

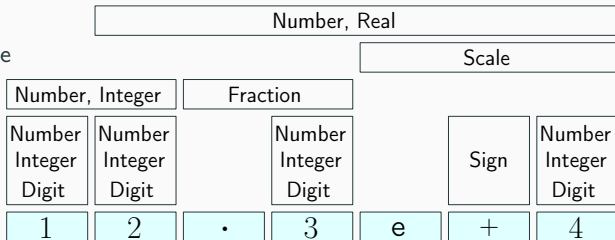
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow e Sign Integer | ϵ

Digit \rightarrow 0 | 1 | ... | 8 | 9

Sign \rightarrow + | -



Example word: 12.3e+4

Number \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

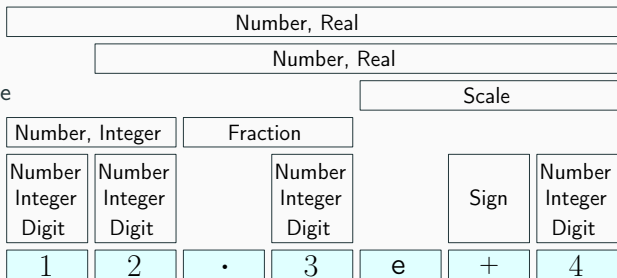
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow e Sign Integer | ϵ

Digit \rightarrow 0 | 1 | ... | 8 | 9

Sign \rightarrow + | -



Example word: 12.3

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 | 1 | \dots | 8 | 9

Sign \rightarrow + | -

| | | | |
|----------------------------|----------------------------|----------|----------------------------|
| Number, Integer | | Fraction | |
| Number Integer Digit | Number Integer Digit | | Number Integer Digit |
| 1 | 2 | . | 3 |

Example word: 12.3

Number \rightarrow Integer | Real

Integer \rightarrow Digit | Integer Digit

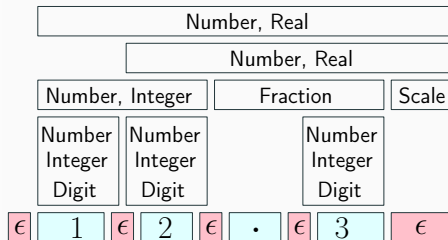
Real \rightarrow Integer Fraction Scale

Fraction \rightarrow . Integer

Scale \rightarrow ϵ Sign Integer | ϵ

Digit \rightarrow 0 | 1 | \dots | 8 | 9

Sign \rightarrow + | -



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

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 $O(n^3|G|)$

Question:

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

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:

$$S \rightarrow AB \mid BC$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow CC \mid b$$

$$C \rightarrow AB \mid a$$

- Input: *ababa*

- 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