CS 3313
Foundations of Computing:

CYK Parsing Algorithm

http://gw-cs3313.github.io

1

Simplification and Parsing

- 1. Simplification rules: transform a grammar such that:
  - Resulting grammar generates the same language
  - and has "more efficient" production rules in a specific format
- 2. Normal Forms: express all CFGs using a standard "format" for how the production rules are specified
  - Definition of CFGs places no restrictions on RHS of production
  - It is convenient (for parsing algorithms) to restrict to a standard form
    - Chomsky Normal Form (CNF) or Greiback Normal Form (GNF)
- 3. Parsing Algorithm: Design a parsing algorithm that takes a grammar in a standard form (CNF) to check if string w is generated by grammar G.

2

Procedure to transform any CFG to Chomsky Normal Form

- A CFG is said to be in *Chomsky Normal Form* if every production is of one of these two forms:
  1. A → BC (right hand side is two variables).
  2. A → a (right hand side is a single terminal).
- Theorem: If L is a CFL, then L – { λ} has a CFG in CNF.
  - *Note: Theorem 2.4 implies every string on RHS of prodution is either a single terminal or has length $\geq$ 2.*
    - *This is our starting point when converting to CNF form*
- Question: property of parse trees for CNF grammars ?

3

3

CNF

- G_1 with production rules:
  - $S \rightarrow AS \mid a$
  - $A \rightarrow SA \mid b$
- Is G_1 in CNF?
- G_2 with production rules:
- P: $S \rightarrow ABa \quad A \rightarrow aab \quad B \rightarrow Ac$
- Is G_2 in CNF?

4

Testing for Membership – a Parsing Algorithm

- Simple algorithm: Convert CFG to a Greibach Normal Form (all productions are of the form $A \rightarrow a\alpha$)
  - For string w of length n, we have n derivation steps.
  - At each step, explore all productions.
  - Time: $O(|P|^n)$ – this is exponential (in length of input string w)

- Example: S → aSB | bSA | aB   A → a   B → b
- w = baba

5

Testing Membership (Parsing)

- Determine if w is in $L(G)$.
- Can we do better than the simple method with GNF with $O(|P|^n)$ for string of length n ?
- Assume G is in CNF.
  - Or convert the given grammar to CNF.
  - w = λ is a special case, solved by testing if the start symbol is nullable.
- Cocke Younger Kashimi Algorithm (CYK) is a good example of dynamic programming and runs in time $O(n^3)$, where $n = |w|$.

6

6

CYK Algorithm notations

- Important: these notations are a bit different from notations in the book, but the end algorithm works in the same manner
- Input string w has length n – i.e, consists of n terminal symbols:

  $w = a_1 a_2 \ldots a_n$ where each $a_i \in T$

  - Ex: $w = abcaab$ $\quad a_1=a \; a_2=b \; a_3= c, \ldots$
- Define a substring x_{ij} *(of w)* as the substring starting at position i and having length j
  - $x_{13} = abc \quad x_{22}= bc \quad x_{33}= caa$
- For a substring x_{ij}, define V_{ij} to be set of variables that derive x_{ij}
  - $V_{ij} = \{ A \mid A \Rightarrow^* x_{ij} \}$
  - Ex: $V_{33} = \{ A \mid A \Rightarrow^* x_{33} = caa \}$

7

Setting up our solution/algorithm: Notations

- Input string $w = abcaab$
- Define a substring x_{ij} *(of w)* as the substring starting at position i and having length j
  - Ex: $x_{13} = abc \quad x_{22}= bc$

    $x_{33}= caa \quad x_{15}= abcaa \quad w=x_{16}=abcaab$
- <u>For a substring x_{ij}, define V_{ij} to be set of variables that derive x_{ij}</u>
  - $V_{ij} = \{ A \mid A \Rightarrow^* x_{ij} \}$

8

Algorithm

- Claim is that we can construct V_{ij} iteratively
- Basis: $V_{i1} = \{ A \mid A \rightarrow x_{i1} \text{ is a production } \}$
- Ind. $A \Rightarrow^* x_{ij}$ iff $A \rightarrow BC$ and for some k, $1 <= k <= j$,

  $B \Rightarrow^* x_{ik}$ and $C \Rightarrow^* x_{i+k,\, j-k}$
- Since k, j-k are $<j$ the IH holds.
- w is in $L(G)$ iff $S \in V_{1n}$ (since $w = x_{1n}$)

$V_{ij} = \{ A \mid A \rightarrow BC$, and

for some k, B is in V_{ik} and C is in $V_{i+k,j-k}$ $\}$

9

CYK Algorithm

Input: CFG G=(V,T,P,S) in CNF, Input string w of length n

1. for i=1 to n

   $V_{i1} = \{A \mid A \rightarrow a \text{ is in } P \text{ and } x_{i1} = a\}$
2. for j=2 to n

   for i=1 to n-j+1 {

   $V_{ij} = \emptyset$

   for k =1 to j-1 {

   $V_{ij} = V_{ij} \cup \{ A \mid A \rightarrow BC$ *is a production in P,*

   B is in V_{ik}

   C is in $V_{i+k,j-1}$ }

   }}
3. w is in L(G) if S is in V_{1n}

10

Time Complexity

- Step 1: takes $O(n)$ to examine each of the n symbols
  - Assume P is a constant.
- Step 2: $O(n^3)$
  - Outer j loop iterates $O(n)$
  - The i loop iterates $O(n)$
  - For each of the n^2 iterations, the k loop iterates $O(n)$
- Dynamic programming formulation
  - Construct solution for size n in terms of sizes $n-1$
    - Principle of optimality needs to hold

11

Example: Application of CYK Algorithm

- $S \rightarrow AB \mid BC$ $\quad A \rightarrow BA \mid a$ $\quad B \rightarrow CC \mid b$ $\quad C \rightarrow AB \mid a$
- $w = baaba$ (length 5), so i,j iterate from 1 to 5.
- Some sample V_{ij}
- To compute V_{31}, $x_{31} = a$. $V_{31} = \{ X \mid X \rightarrow a \text{ is in } P\}$
  - $V_{31} = \{ A, C\}$
- To compute V_{12}: $X \rightarrow YZ$ *in* P and
  - check if $Y \epsilon V_{11}$ and $Z \epsilon V_{21}$
- To compute V_{23} : $X \rightarrow YZ$ *in* P and
  - Check for Y *in* V_{21} and Z *in* V_{32}
  - Check for Y *in* V_{22} and Z *in* V_{41}

12

Example: Application of CYK Algorithm

- S →AB | BC   A→ BA| a   B → CC |b   C→ AB| a
- *w = baaba* (length 5), so *i,j* iterate from 1 to 5.
- $1 \leq k \leq j-1$

Visualize computation as a 2-D n by n array

i →

j ↓

	1	2	3	4	5
1	V_{11}	V_{11}	V_{11}	V_{11}	V_{51}
2	V_{12}	V_{22}	V_{32}	V_{42}	
3	V_{13}	V_{23}	V_{33}		
4	V_{14}	V_{24}			
5	V_{15}				

13

Example: Application of CYK Algorithm

- S →AB | BC   A→ BA| a   B → CC |b   C→ AB| a
- *w = baaba* (length 5), so *i,j* iterate from 1 to 5.

Visualize computation as a 2-D n by n array

i →

j ↓

i=1 j=2				
i=1 j=5				

14

Example: Application of CYK Algorithm

- S →AB | BC   A→ BA| a   B → CC |b   C→ AB| a
- *w = baaba* (length 5), so *i,j* iterate from 1 to 5
- $V_{12} = \{ X \mid X \rightarrow YZ,\ Y \text{ in } V_{11} = (B) \text{ and } Z \text{ in } V_{21} = (A,C) \}$

Visualize computation as a 2-D n by n array

j \ *i*	1 $x_{11}=b$	2 $x_{21}=a$	3 $x_{31}=a$	4 $x_{41}=b$	5 $x_{51}=a$
1	$V_{11} = \{B\}$	$V_{21} = \{A,C\}$	$V_{31} = \{A,C\}$	$V_{41} = \{B\}$	$V_{51} = \{A,C\}$
2	$V_{12} = \{S,A\}$				
3					
4					
5	V_{15}				

$V_{51} = \{B\}$

15

Example: CYK Algorithm

- S →AB | BC   A→ BA| a   B → CC |b   C→ AB| a
- $V_{24} = \{ X \mid X \rightarrow YZ,\ Y \text{ in } V_{21} \text{ and } Z \text{ in } V_{33} \} \cup \{ X \mid X \rightarrow YZ,\ Y \text{ in } V_{22} \text{ and } Z \text{ in } V_{42} \}$
  $\cup \{ X \mid X \rightarrow YZ,\ Y \text{ in } V_{23} \text{ and } Z \text{ in } V_{51} \}$

j \ *i*	1 $x_{11}=b$	2 $x_{21}=a$	3 $x_{31}=a$	4 $x_{41}=b$	5 $x_{51}=a$
1	B	**A,C**	A,C	B	**A,C**
2	S,A	**B**	S,C	**S,A**	
3	Ø	**B**	**B**		
4	Ø				
5	V_{15}				

16

Application of CYK Algorithm

- S →AB | BC   A→ BA| a   B → CC |b   C→ AB| a
- $w_1 = baaba$
- $w_2 = aab$?

B	A, C	A, C	B	A, C
S, A	B	S, C	A, S	
∅	B	B		
∅	B			
S, A, C				

S is in V_{15} therefore w is in L(G)

17

Application of CYK Algorithm

- S →AB | BC   A→ BA| a   B → CC |b   C→ AB| a
- *Parse tree for* $w_1 = baaba$

B	A, C	A, C	B	A, C
S, A	B	S, C	A, S	
∅	B	B		
∅	B			
S, A, C				

S is in V_{15} therefore w is in L(G)

18