CSE 135: Introduction to Theory of Computation
CFLs: Closure Properties and Membership Test

Sungjin Im

University of California, Merced

03-31-2014

Union of CFLs

Let L_1 be language recognized by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language recognized by $G_2 = (V_2, \Sigma_2, R_2, S_2)$
Is $L_1 \cup L_2$ a context free language?

Union of CFLs

Let L_1 be language recognized by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language recognized by $G_2 = (V_2, \Sigma_2, R_2, S_2)$
Is $L_1 \cup L_2$ a context free language? Yes.

Union of CFLs

Let L_1 be language recognized by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language recognized by $G_2 = (V_2, \Sigma_2, R_2, S_2)$
Is $L_1 \cup L_2$ a context free language? Yes.
Just add the rule $S \rightarrow S_1 | S_2$

Union of CFLs

Let L_1 be language recognized by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language recognized by $G_2 = (V_2, \Sigma_2, R_2, S_2)$
Is $L_1 \cup L_2$ a context free language? Yes.
Just add the rule $S \to S_1 | S_2$
But make sure that $V_1 \cap V_2 = \emptyset$ (by renaming some variables).

Union of CFLs

Let L_1 be language recognized by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language recognized by $G_2 = (V_2, \Sigma_2, R_2, S_2)$
Is $L_1 \cup L_2$ a context free language? Yes.
Just add the rule $S \to S_1 | S_2$
But make sure that $V_1 \cap V_2 = \emptyset$ (by renaming some variables).

Closure of CFLs under Union

$G = (V, \Sigma, R, S)$ such that $L(G) = L(G_1) \cup L(G_2)$:

- $V = V_1 \cup V_2 \cup \{S\}$ (the three sets are disjoint)
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $R = R_1 \cup R_2 \cup \{S \to S_1 | S_2\}$

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation:

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation: $L_1 L_2$ generated by a grammar with an additional rule $S \to S_1 S_2$

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation: $L_1 L_2$ generated by a grammar with an additional rule $S \to S_1 S_2$
- Kleene Closure:

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation: $L_1 L_2$ generated by a grammar with an additional rule $S \to S_1 S_2$
- Kleene Closure: L_1^* generated by a grammar with an additional rule $S \to S_1 S | \epsilon$

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation: $L_1 L_2$ generated by a grammar with an additional rule $S \rightarrow S_1 S_2$
- Kleene Closure: L_1^* generated by a grammar with an additional rule $S \rightarrow S_1 S | \epsilon$

As before, ensure that $V_1 \cap V_2 = \emptyset$. S is a new start symbol.

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation: $L_1 L_2$ generated by a grammar with an additional rule $S \to S_1 S_2$
- Kleene Closure: L_1^* generated by a grammar with an additional rule $S \to S_1 S | \epsilon$

As before, ensure that $V_1 \cap V_2 = \emptyset$. S is a new start symbol. (Exercise: Complete the Proof!) $\square$

Intersection

Let L_1 and L_2 be context free languages.

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL
  - Generated by a grammar with rules $S \to XY$; $X \to aXb|\epsilon$; $Y \to cY|\epsilon$.

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aXb|\epsilon$; $Y \rightarrow cY|\epsilon$.
- $L_2 = \{a^i b^j c^j \mid i, j \geq 0\}$ is a CFL.

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aXb|\epsilon$; $Y \rightarrow cY|\epsilon$.
- $L_2 = \{a^i b^j c^j \mid i, j \geq 0\}$ is a CFL.
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aX|\epsilon$; $Y \rightarrow bYc|\epsilon$.

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aXb|\epsilon$; $Y \rightarrow cY|\epsilon$.
- $L_2 = \{a^i b^j c^j \mid i, j \geq 0\}$ is a CFL.
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aX|\epsilon$; $Y \rightarrow bYc|\epsilon$.
- But $L_1 \cap L_2 =$

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aXb|\epsilon$; $Y \rightarrow cY|\epsilon$.
- $L_2 = \{a^i b^j c^j \mid i, j \geq 0\}$ is a CFL.
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aX|\epsilon$; $Y \rightarrow bYc|\epsilon$.
- But $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ is not a CFL. $\square$

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R.

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M)

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M): $Q_{P'} = Q_P \times Q_M$

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M): $Q_{P'} = Q_P \times Q_M$
- These determine the transition function of P'.

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M): $Q_{P'} = Q_P \times Q_M$
- These determine the transition function of P'.
- The final states of P' are those in which both the state of P and state of M are accepting:

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M): $Q_{P'} = Q_P \times Q_M$
- These determine the transition function of P'.
- The final states of P' are those in which both the state of P and state of M are accepting: $F_{P'} = F_P \times F_M$ $\square$

Intersection with Regular Languages

Proposition

If L is a CFL and R is a regular language then $L \cap R$ is a CFL.

Proof.

Let P be the PDA that accepts L, and let M be the DFA that accepts R. A new PDA P' will simulate P and M simultaneously on the same input and accept if both accept. Then P' accepts $L \cap R$.

- The stack of P' is the stack of P
- The state of P' at any time is the pair (state of P, state of M): $Q_{P'} = Q_P \times Q_M$
- These determine the transition function of P'.
- The final states of P' are those in which both the state of P and state of M are accepting: $F_{P'} = F_P \times F_M$ □

Why does this construction not work for intersection of two CFLs?

Complementation

Let L be a context free language. Is $\overline{L}$ context free?

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation.

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL.

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL.

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! $\square$

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! $\square$

Proof 2.

$L = \{x \mid x \text{ not of the form } ww\}$ is a CFL.

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! $\square$

Proof 2.

$L = \{x \mid x \text{ not of the form } ww\}$ is a CFL.

- L generated by a grammar with rules

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! □

Proof 2.

$L = \{x \mid x \text{ not of the form } ww\}$ is a CFL.

- L generated by a grammar with rules $X \to a|b$, $A \to a|XAX$, $B \to b|XBX$, $S \to$

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! $\square$

Proof 2.

$L = \{x \mid x \text{ not of the form } ww\}$ is a CFL.

- L generated by a grammar with rules $X \rightarrow a|b$, $A \rightarrow a|XAX$, $B \rightarrow b|XBX$, $S \rightarrow A|B|AB|BA$

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.
Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! $\square$

Proof 2.
$L = \{x \mid x \text{ not of the form } ww\}$ is a CFL.

- L generated by a grammar with rules $X \to a|b$, $A \to a|XAX$, $B \to b|XBX$, $S \to A|B|AB|BA$

But $\overline{L} = \{ww \mid w \in \{a,b\}^*\}$ is not a CFL! (Why?) $\square$

Set Difference

Proposition

If L_1 is a CFL and L_2 is a CFL then $L_1 \setminus L_2$ is not necessarily a CFL

Set Difference

Proposition

If L_1 is a CFL and L_2 is a CFL then $L_1 \setminus L_2$ is not necessarily a CFL

Proof.

Because CFLs not closed under complementation, and complementation is a special case of set difference. (How?) $\square$

Set Difference

Proposition

If L_1 is a CFL and L_2 is a CFL then $L_1 \setminus L_2$ is not necessarily a CFL

Proof.

Because CFLs not closed under complementation, and complementation is a special case of set difference. (How?) □

Proposition

If L is a CFL and R is a regular language then $L \setminus R$ is a CFL

Set Difference

Proposition

If L_1 is a CFL and L_2 is a CFL then $L_1 \setminus L_2$ is not necessarily a CFL

Proof.

Because CFLs not closed under complementation, and complementation is a special case of set difference. (How?) $\square$

Proposition

If L is a CFL and R is a regular language then $L \setminus R$ is a CFL

Proof.

$L \setminus R = L \cap \overline{R}$ $\square$

Emptiness Problem

Given a CFG G with start symbol S, is $L(G)$ empty?

Emptiness Problem

Given a CFG G with start symbol S, is $L(G)$ empty?

Solution: Check if the start symbol S is generating.

Emptiness Problem

Given a CFG G with start symbol S, is $L(G)$ empty?
Solution: Check if the start symbol S is generating. How long does that take?

Determining generating symbols

Algorithm

```
Gen = {}
for every rule A → x where x ∈ Σ*
    Gen = Gen ∪ {A}
repeat
    for every rule A → γ
        if all variables in γ are generating then
            Gen = Gen ∪ {A}
until Gen does not change
```

Determining generating symbols

Algorithm

```
Gen = {}
for every rule A → x where x ∈ Σ*
    Gen = Gen ∪ {A}
repeat
    for every rule A → γ
        if all variables in γ are generating then
            Gen = Gen ∪ {A}
until Gen does not change
```

- Both for-loops take $O(n)$ time where $n = |G|$.

Determining generating symbols

Algorithm

```
Gen = {}
for every rule A → x where x ∈ Σ*
    Gen = Gen ∪ {A}
repeat
    for every rule A → γ
        if all variables in γ are generating then
            Gen = Gen ∪ {A}
until Gen does not change
```

- Both for-loops take $O(n)$ time where $n = |G|$.
- Each iteration of repeat-until loop discovers a new variable. So number of iterations is $O(n)$. And total is $O(n^2)$.

Membership Problem

Given a CFG $G = (V, \Sigma, R, S)$ in Chomsky Normal Form, and a string $w \in \Sigma^*$, is $w \in L(G)$?

Membership Problem

Given a CFG $G = (V, \Sigma, R, S)$ in Chomsky Normal Form, and a string $w \in \Sigma^*$, is $w \in L(G)$?
Central question in parsing.

“Simple” Solution

“Simple” Solution

- Let $|w| = n$. Since G is in Chomsky Normal Form, w has a parse tree of size $2n - 1$ iff $w \in L(G)$

"Simple" Solution

- Let $|w| = n$. Since G is in Chomsky Normal Form, w has a parse tree of size $2n - 1$ iff $w \in L(G)$
- Construct all possible parse (binary) trees and check if any of them is a valid parse tree for w

“Simple” Solution

- Let $|w| = n$. Since G is in Chomsky Normal Form, w has a parse tree of size $2n - 1$ iff $w \in L(G)$
- Construct all possible parse (binary) trees and check if any of them is a valid parse tree for w
- Number of parse trees of size $2n - 1$ is k^{2n-1} where k is the number of variables in G. So algorithm is exponential in n!

"Simple" Solution

- Let $|w| = n$. Since G is in Chomsky Normal Form, w has a parse tree of size $2n-1$ iff $w \in L(G)$
- Construct all possible parse (binary) trees and check if any of them is a valid parse tree for w
- Number of parse trees of size $2n-1$ is k^{2n-1} where k is the number of variables in G. So algorithm is exponential in n!
- We will see an algorithm that runs in $O(n^3)$ time (the constant will depend on k).

First Ideas

Notation

Suppose $w = w_1 w_2 \cdots w_n$, where $w_i \in \Sigma$. Let $w_{i,j}$ denote the substring of w starting at position i of length j. Thus, $w_{i,j} = w_i w_{i+1} \cdots w_{i+j-1}$

First Ideas

Notation

Suppose $w = w_1 w_2 \cdots w_n$, where $w_i \in \Sigma$. Let $w_{i,j}$ denote the substring of w starting at position i of length j. Thus, $w_{i,j} = w_i w_{i+1} \cdots w_{i+j-1}$

Main Idea

For every $A \in V$, and every $i \leq n$, $j \leq n + 1 - i$, we will determine if $A \stackrel{*}{\Rightarrow} w_{i,j}$.

First Ideas

Notation

Suppose $w = w_1 w_2 \cdots w_n$, where $w_i \in \Sigma$. Let $w_{i,j}$ denote the substring of w starting at position i of length j. Thus, $w_{i,j} = w_i w_{i+1} \cdots w_{i+j-1}$

Main Idea

For every $A \in V$, and every $i \leq n$, $j \leq n+1-i$, we will determine if $A \stackrel{*}{\Rightarrow} w_{i,j}$.

Now, $w \in L(G)$ iff $S \stackrel{*}{\Rightarrow} w_{1,n} = w$; thus, we will solve the membership problem.

First Ideas

Notation

Suppose $w = w_1 w_2 \cdots w_n$, where $w_i \in \Sigma$. Let $w_{i,j}$ denote the substring of w starting at position i of length j. Thus, $w_{i,j} = w_i w_{i+1} \cdots w_{i+j-1}$

Main Idea

For every $A \in V$, and every $i \leq n$, $j \leq n + 1 - i$, we will determine if $A \stackrel{*}{\Rightarrow} w_{i,j}$.

Now, $w \in L(G)$ iff $S \stackrel{*}{\Rightarrow} w_{1,n} = w$; thus, we will solve the membership problem.

How do we determine if $A \stackrel{*}{\Rightarrow} w_{i,j}$ for every A, i, j?

Base Case

Substrings of length 1

Observation

For any A, i, $A \overset{*}{\Rightarrow} w_{i,1}$ iff $A \rightarrow w_{i,1}$ is a rule.

Base Case

Substrings of length 1

Observation

For any A, i, $A \stackrel{*}{\Rightarrow} w_{i,1}$ iff $A \rightarrow w_{i,1}$ is a rule.

- Since G is in Chomsky Normal Form, G does not have any ϵ-rules, nor any unit rules.

Base Case

Substrings of length 1

Observation

For any A, i, $A \stackrel{*}{\Rightarrow} w_{i,1}$ iff $A \rightarrow w_{i,1}$ is a rule.

- Since G is in Chomsky Normal Form, G does not have any ϵ-rules, nor any unit rules.

Thus, for each A and i, one can determine if $A \stackrel{*}{\Rightarrow} w_{i,1}$.

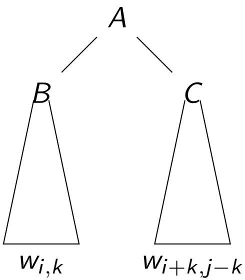
Inductive Step

Longer substrings

Suppose for every variable X and every $w_{i,\ell}$ ($\ell < j$) we have determined if $X \overset{*}{\Rightarrow} w_{i,\ell}$

Inductive Step

Longer substrings

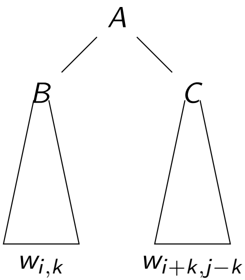


Suppose for every variable X and every $w_{i,\ell}$ ($\ell < j$) we have determined if $X \stackrel{*}{\Rightarrow} w_{i,\ell}$

- $A \stackrel{*}{\Rightarrow} w_{i,j}$ iff there are variables B and C and some $k < j$ such that $A \rightarrow BC$ is a rule, and $B \stackrel{*}{\Rightarrow} w_{i,k}$ and $C \stackrel{*}{\Rightarrow} w_{i+k,j-k}$

Inductive Step

Longer substrings



Suppose for every variable X and every $w_{i,\ell}$ ($\ell < j$) we have determined if $X \stackrel{*}{\Rightarrow} w_{i,\ell}$

- $A \stackrel{*}{\Rightarrow} w_{i,j}$ iff there are variables B and C and some $k < j$ such that $A \to BC$ is a rule, and $B \stackrel{*}{\Rightarrow} w_{i,k}$ and $C \stackrel{*}{\Rightarrow} w_{i+k,j-k}$
- Since k and $j - k$ are both less than j, we can inductively determine if $A \stackrel{*}{\Rightarrow} w_{i,j}$.

Cocke-Younger-Kasami (CYK) Algorithm

Algorithm maintains $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$.

```
Initialize:   X_{i,1} = {A | A → w_{i,1}}
for j = 2 to n do
    for i = 1 to n − j + 1 do
        X_{i,j} = ∅
        for k = 1 to j − 1 do
            X_{i,j} = X_{i,j} ∪ {A | A → BC, B ∈ X_{i,k}, C ∈ X_{i+k,j−k}}
```

Cocke-Younger-Kasami (CYK) Algorithm

Algorithm maintains $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$.

```
Initialize:   X_{i,1} = {A | A → w_{i,1}}
for j = 2 to n do
    for i = 1 to n − j + 1 do
        X_{i,j} = ∅
        for k = 1 to j − 1 do
            X_{i,j} = X_{i,j} ∪ {A | A → BC, B ∈ X_{i,k}, C ∈ X_{i+k,j−k}}
```

Correctness: After each iteration of the outermost loop, $X_{i,j}$ contains exactly the set of variables A that can derive $w_{i,j}$, for each i.

Cocke-Younger-Kasami (CYK) Algorithm

Algorithm maintains $X_{i,j} = \{A \mid A \stackrel{*}{\Rightarrow} w_{i,j}\}$.

```
Initialize:   X_{i,1} = {A | A → w_{i,1}}
for j = 2 to n do
    for i = 1 to n − j + 1 do
        X_{i,j} = ∅
        for k = 1 to j − 1 do
            X_{i,j} = X_{i,j} ∪ {A | A → BC, B ∈ X_{i,k}, C ∈ X_{i+k,j−k}}
```

Correctness: After each iteration of the outermost loop, $X_{i,j}$ contains exactly the set of variables A that can derive $w_{i,j}$, for each i.Time $= O(n^3)$.

Example

Example

Consider grammar
$S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a$ Let
$w = baaba$. The sets $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$:

Example

Example

Consider grammar
$S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a$ Let
$w = baaba$. The sets $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$:

j/i	1	2	3	4	5
5					
4					
3					
2					
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
	b	a	a	b	a

Example

Example

Consider grammar
$S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a$ Let
$w = baaba$. The sets $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$:

j/i	1	2	3	4	5
5					
4					
3					
2	$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
	b	a	a	b	a

Example

Example

Consider grammar
$S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a$ Let
$w = baaba$. The sets $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$:

j/i	1	2	3	4	5
5					
4					
3	$\emptyset$	$\{B\}$	$\{B\}$		
2	$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
	b	a	a	b	a

Example

Example

Consider grammar
$S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a$ Let
$w = baaba$. The sets $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$:

j/i	1	2	3	4	5
5					
4	$\emptyset$	$\{S, A, C\}$			
3	$\emptyset$	$\{B\}$	$\{B\}$		
2	$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
	b	a	a	b	a

Example

Example

Consider grammar
$S \to AB \mid BC,\ A \to BA \mid a,\ B \to CC \mid b,\ C \to AB \mid a$ Let
$w = baaba$. The sets $X_{i,j} = \{A \mid A \overset{*}{\Rightarrow} w_{i,j}\}$:

j/i	1	2	3	4	5
5	$\{S, A, C\}$				
4	$\emptyset$	$\{S, A, C\}$			
3	$\emptyset$	$\{B\}$	$\{B\}$		
2	$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
	b	a	a	b	a

More Decision Problems

Given a CFGs G_1 and G_2

More Decision Problems

Given a CFGs G_1 and G_2

- Is $L(G_1) = \Sigma^*$?

More Decision Problems

Given a CFGs G_1 and G_2

- Is $L(G_1) = \Sigma^*$?
- Is $L(G_1) \cap L(G_2) = \emptyset$?

More Decision Problems

Given a CFGs G_1 and G_2

- Is $L(G_1) = \Sigma^*$?
- Is $L(G_1) \cap L(G_2) = \emptyset$?
- Is $L(G_1) = L(G_2)$?

More Decision Problems

Given a CFGs G_1 and G_2

- Is $L(G_1) = \Sigma^*$?
- Is $L(G_1) \cap L(G_2) = \emptyset$?
- Is $L(G_1) = L(G_2)$?
- Is G_1 ambiguous?

More Decision Problems

Given a CFGs G_1 and G_2

- Is $L(G_1) = \Sigma^*$?
- Is $L(G_1) \cap L(G_2) = \emptyset$?
- Is $L(G_1) = L(G_2)$?
- Is G_1 ambiguous?
- Is $L(G_1)$ inherently ambiguous?

More Decision Problems

Given a CFGs G_1 and G_2

- Is $L(G_1) = \Sigma^*$?
- Is $L(G_1) \cap L(G_2) = \emptyset$?
- Is $L(G_1) = L(G_2)$?
- Is G_1 ambiguous?
- Is $L(G_1)$ inherently ambiguous?

All these problems are undecidable.