Ogden's Lemma for CFLs

Theorem

If L is a context-free language, then there exists an integer l such that for any $u \in L$ with at least l positions marked, u can be written as $u = vwxyz$ such that

1. *x and at least one of w or y both contain a marked position;*
2. *wxy contains at most l marked positions; and,*
3. *$vw^m xy^m z \in L$ for all $m \in N$.*

Consider language $\{a^i b^j c^k d^l \mid i = 0 \ \ or \ \ j = k = l\}$, for which the classical PL fails (why?).

Non-Decision Properties

- Many questions that can be decided for regular sets cannot be decided for CFLs.
- Example: Are two CFLs the same?
- Example: Are two CFLs disjoint?
- Need theory of Turing machines and decidability to prove no algorithm exists.

Testing Emptiness

- We already did this.
- We learned to eliminate variables that generate no terminal string.
- If the start symbol is one of these, then the CFL is empty; otherwise not.

Testing Membership

- Want to know if string w is in $L(G)$.
- Assume G is in CNF.
  - Or convert the given grammar to CNF.
  - $w = \epsilon$ is a special case, solved by testing if the start symbol is nullable.
- Algorithm (CYK) is a good example of dynamic programming and runs in time $O(n^3)$, where $n = |w|$.

CYK Algorithm

- Let $w = a_1 ... a_n$.
- We construct an n-by-n triangular array of sets of variables.
- $X_{ij} = \{\text{variables } A \mid A \overset{*}{\Rightarrow} a_i ... a_j\}$.
- Induction on $j - i + 1$. The length of the derived string.
- Finally, ask if S is in X_{1n}.

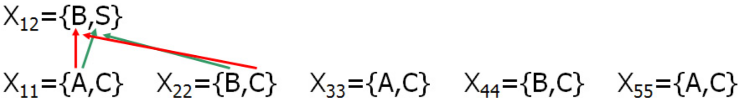
CYK Algorithm V (2)

- Basis: $X_{ii} = \{A \mid A \to a_i$ is a production $\}$.
- Induction: $X_{ij} = \{A \mid$ there is a production $A \to BC$ and an integer $k, i < k < j, B \in X_{ik}, C \in X_{k+1,j}\}$.

Example

Grammar: $S \to AB, \quad A \to BC \mid a, \quad B \to AC \mid b, \quad C \to a \mid b$
String $w = ababa$

X_{11}={A,C} X_{22}={B,C} X_{33}={A,C} X_{44}={B,C} X_{55}={A,C}



Example (cont'd)

Example

Grammar: $S \rightarrow AB, \quad A \rightarrow BC \mid a, \quad B \rightarrow AC \mid b, \quad C \rightarrow a \mid b$

String $w = ababa$

X_{13}={} Yields nothing

X_{12}={B,S} X_{23}={A} X_{34}={B,S} X_{45}={A}

X_{11}={A,C} X_{22}={B,C} X_{33}={A,C} X_{44}={B,C} X_{55}={A,C}

X_{13}={A} X_{24}={B,S} X_{35}={A}

X_{12}={B,S} X_{23}={A} X_{34}={B,S} X_{45}={A}

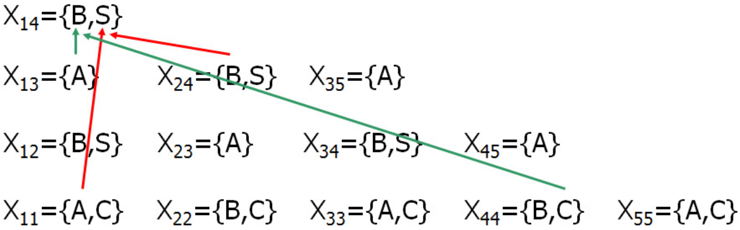
X_{11}={A,C} X_{22}={B,C} X_{33}={A,C} X_{44}={B,C} X_{55}={A,C}

Example (cont'd)

Example

Grammar: $S \rightarrow AB, \quad A \rightarrow BC \mid a, \quad B \rightarrow AC \mid b, \quad C \rightarrow a \mid b$

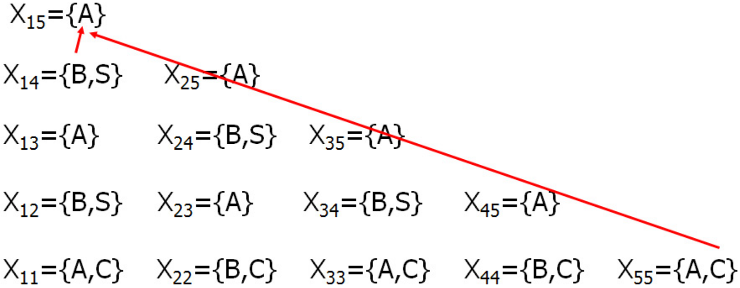
String $w = ababa$



Example (cont'd)

Example

Grammar: $S \to AB, \quad A \to BC \mid a, \quad B \to AC \mid b, \quad C \to a \mid b$
String $w = ababa$



Testing Infiniteness

- The idea is essentially the same as for regular languages.
- Use the pumping lemma constant n.
- If there is a string in the language of length between n and $2n - 1$, then the language is infinite; otherwise not.
- Lets work this out in class.

Closure Properties of CFLs

- CFLs are closed under union, concatenation, and Kleene closure.
- Also, under reversal, homomorphisms and inverse homomorphisms.
- But not under intersection or difference.

Closure of CFLs Under Reversal

- If L is a CFL with grammar G, form a grammar for L^R by reversing the right side of every production.
- Example: Let G have $S \rightarrow 0S1 \mid 01$.
- The reversal of $L(G)$ has grammar $S \rightarrow 1S0 \mid 10$.

Closure of CFLs Under Homomorphism

- Let L be a CFL with grammar G.
- Let h be a homomorphism on the terminal symbols of G.
- Construct a grammar for $h(L)$ by replacing each terminal symbol a by $h(a)$.

Example

G has productions $S \rightarrow 0S1 \mid 01$. h is defined by $h(0) = ab, h(1) = \epsilon$.
$h(L(G))$ has the grammar with productions $S \rightarrow abS \mid ab$.

Closure of CFLs Under Inverse Homomorphism

- Here, grammars don't help us.
- But a PDA construction serves nicely.
- Intuition: Let $L = L(P)$ for some PDA P.
- Construct PDA P' to accept $h^{-1}(L)$.
- P' simulates P, but keeps, as one component of a two-component state a buffer that holds the result of applying h to one input symbol.

Construction of P'

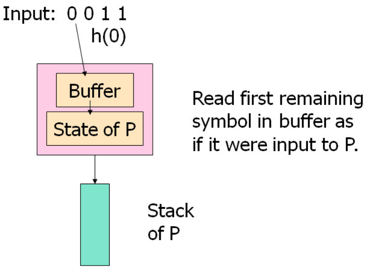
- States are pairs $[q, b]$, where:
  1. q is a state of P.
  2. b is a suffix of $h(a)$ for some symbol a.

  Thus, only a finite number of possible values for b.

- Stack symbols of P' are those of P.

- Start state of P' is $[q_0, \epsilon]$.
- Input symbols of P' are the symbols to which h applies.

- Final states of P' are the states $[q, \epsilon]$ such that q is a final state of P.

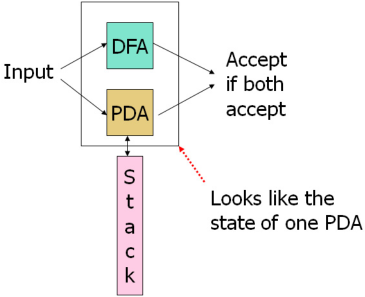


Transitions of P'

1. $\delta'(([q,\epsilon], a, X) = \{([q, h(a)], X)\}$ for any input symbol a of P' and any stack symbol X.
   - When the buffer is empty, P' can reload it.
2. $\delta'([q, bw], \epsilon, X)$ contains $([p, w], \alpha)$ if $\delta(q, b, X)$ contains (p, α), where b is either an input symbol of P or ϵ.
   - Simulate P from the buffer.

Intersection with a Regular Language

- Intersection of two CFL's need not be context free.
- But the intersection of a CFL with a regular language is always a CFL.
- Proof involves running a DFA in parallel with a PDA, and noting that the combination is a PDA. (PDAs accept by final state.)



Formal Construction

- Let the DFA A have transition function δ_A.
- Let the PDA P have transition function δ_P.
- States of combined PDA are $[q, p]$, where q is a state of A and p a state of P.
- $\delta([q, p], a, X)$ contains $([\delta_A(q, a), r], \alpha)$ if $\delta_P(p, a, X)$ contains (r, α). Note a could be ϵ, in which case $\delta_A(q, a) = q$.
- Accepting states of combined PDA are those $[q, p]$ such that q is an accepting state of A and p is an accepting state of P.
- Easy induction: $([q_0, p_0], w, Z_0) \overset{*}{\vdash} ([q, p], \epsilon, \alpha)$ if and only if $\delta_A(q_0, w) = q$ and in $P : (p_0, w, Z_0) \overset{*}{\vdash} (p, \epsilon, \alpha)$.