Properties of Context-Free Languages

Decision Properties

Closure Properties

Summary of Decision Properties

- As usual, when we talk about "a CFL" we really mean "a representation for the CFL, e.g., a CFG or a PDA accepting by final state or empty stack.
- There are algorithms to decide if:
  1. String w is in CFL L.
  2. CFL L is empty.
  3. CFL L is infinite.

Non-Decision Properties

- Many questions that can be decided for regular sets cannot be decided for CFL's.
- Example: Are two CFL's the same?
- Example: Are two CFL's disjoint?
  - How would you do that for regular languages?
- Need theory of Turing machines and decidability to prove no algorithm exists.

Testing Emptiness

- We already did this.
- We learned to eliminate useless variables.
- 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 $w = a_1 \ldots a_n$.
- We construct an n-by-n triangular array of sets of variables.
- X_{ij} = {variables A | A =>* $a_i \ldots a_j$}.
- Induction on $j-i+1$.
  - The length of the derived string.
- Finally, ask if S is in X_{1n}.

CYK Algorithm – (2)

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

Example: CYK Algorithm

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa

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

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

Example: CYK Algorithm

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa

$X_{13}=\{\}$ Yields nothing

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

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

Example: CYK Algorithm

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa

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: CYK Algorithm

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa

X_{14}={B,S}

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: CYK Algorithm

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa

X_{15}={A}

X_{14}={B,S} X_{25}={A}

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}

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.

Closure Properties of CFL's

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

Closure of CFL's Under Union

- Let L and M be CFL's with grammars G and H, respectively.
- Assume G and H have no variables in common.
  - Names of variables do not affect the language.
- Let S_1 and S_2 be the start symbols of G and H.

Closure Under Union – (2)

- Form a new grammar for $L \cup M$ by combining all the symbols and productions of G and H.
- Then, add a new start symbol S.
- Add productions $S \rightarrow S_1 \mid S_2$.

Closure Under Union – (3)

- In the new grammar, all derivations start with S.
- The first step replaces S by either S_1 or S_2.
- In the first case, the result must be a string in L(G) = L, and in the second case a string in L(H) = M.

Closure of CFL's Under Concatenation

- Let L and M be CFL's with grammars G and H, respectively.
- Assume G and H have no variables in common.
- Let S_1 and S_2 be the start symbols of G and H.

Closure Under Concatenation – (2)

- Form a new grammar for LM by starting with all symbols and productions of G and H.
- Add a new start symbol S.
- Add production S -> S_1S_2.
- Every derivation from S results in a string in L followed by one in M.

Closure Under Star

- Let L have grammar G, with start symbol S_1.
- Form a new grammar for L* by introducing to G a new start symbol S and the productions $S \to S_1S \mid \epsilon$.
- A rightmost derivation from S generates a sequence of zero or more S_1's, each of which generates some string in L.

Closure of CFL's Under Reversal

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

Closure of CFL's 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).

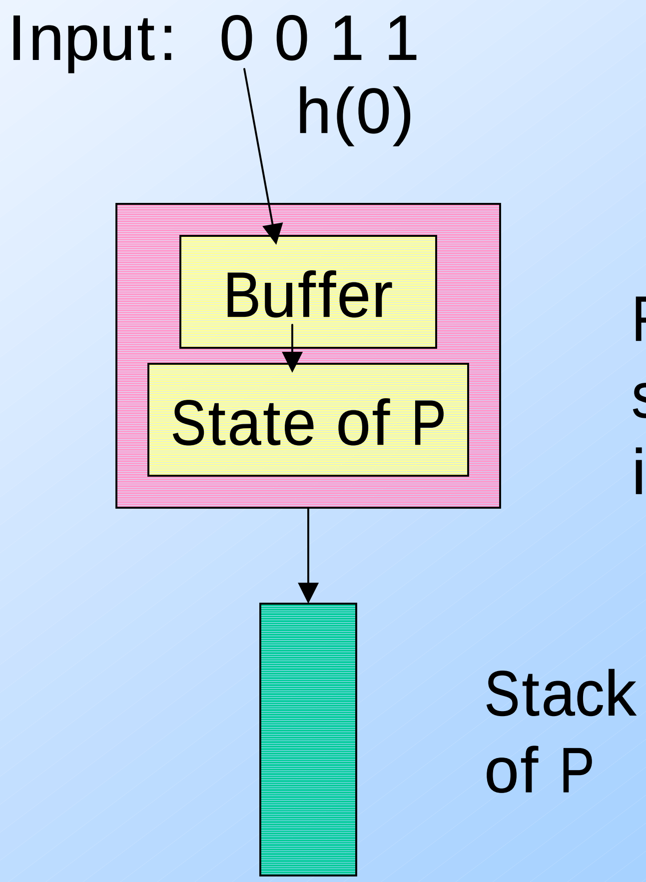
Example: Closure Under Homomorphism

- G has productions S -> 0S1 | 01.
- h is defined by h(0) = ab, h(1) = ϵ.
- h(L(G)) has the grammar with productions S -> abS | ab.

Closure of CFL's Under Inverse Homomorphism

- Here, grammars don't help us, but a PDA construction serves nicely.
- 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.

Architecture of P'



Read first remaining symbol in buffer as if it were input to P.

Formal Construction of P’

- States are pairs [q, w], where:
  1. q is a state of P.
  2. w is a suffix of h(a) for some symbol *a*.
     - Thus, only a finite number of possible values for w.
- Stack symbols of P’ are those of P.
- Start state of P’ is $[q_0, \epsilon]$.

Construction of P' – (2)

- Input symbols of P' are the symbols to which h applies.
- Final states of P' are the states [q, ϵ] 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.

Proving Correctness of P'

- We need to show that L(P') = h^{-1}(L(P)).
- Key argument: P' makes the transition $([q_0, \epsilon], w, Z_0) \vdash^* ([q, x], \epsilon, \alpha)$ if and only if P makes transition $(q_0, y, Z_0) \vdash^* (q, \epsilon, \alpha)$, h(w) = yx, and x is a suffix of the last symbol of w.
- Proof in both directions is an induction on the number of moves made.

Nonclosure Under Intersection

- Unlike the regular languages, the class of CFL's is not closed under $\cap$.
- We know that $L_1 = \{0^n1^n2^n \mid n \geq 1\}$ is not a CFL (use the pumping lemma).
- However, $L_2 = \{0^n1^n2^i \mid n \geq 1, i \geq 1\}$ is.
  - CFG: S -> AB, A -> 0A1 | 01, B -> 2B | 2.
- So is $L_3 = \{0^i1^n2^n \mid n \geq 1, i \geq 1\}$.
- But $L_1 = L_2 \cap L_3$.

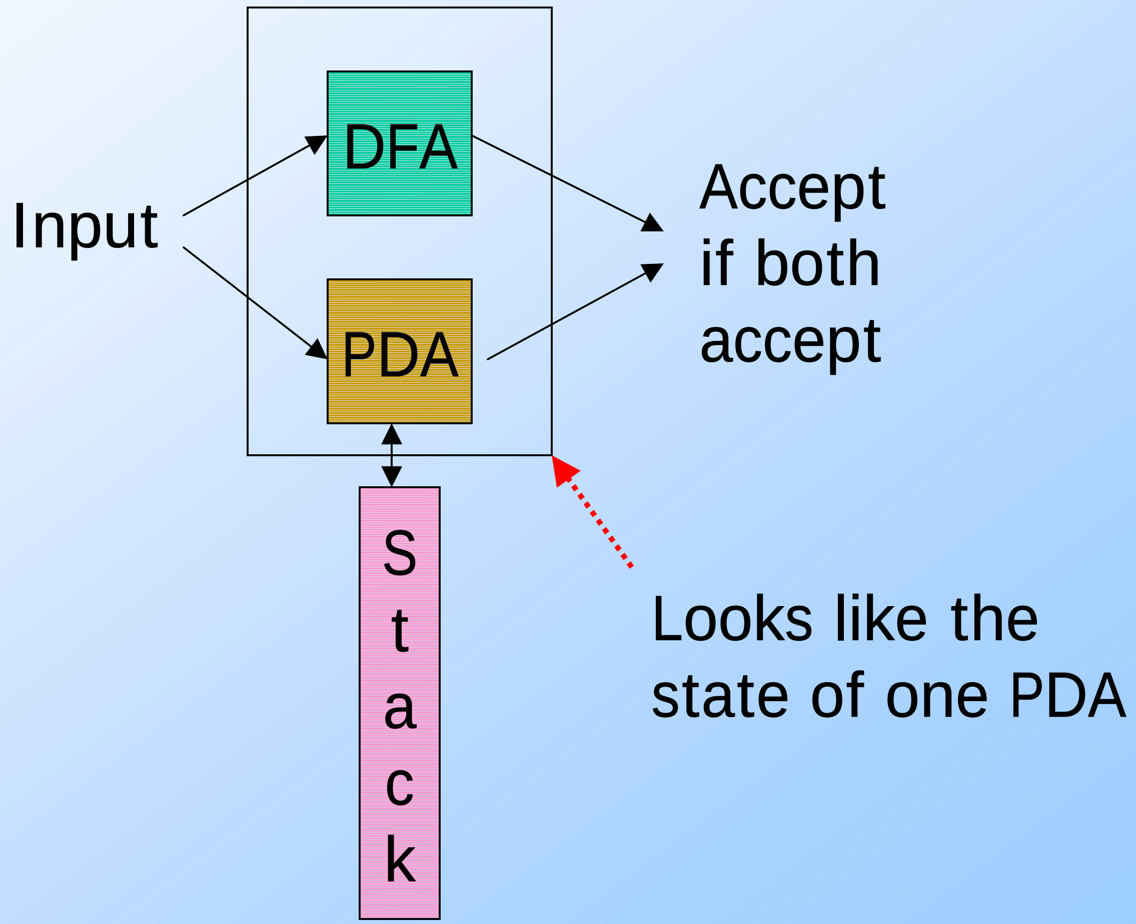
Nonclosure Under Difference

- We can prove something more general:
  - Any class of languages that is closed under difference is closed under intersection.
- Proof: $L \cap M = L - (L - M)$.
- Thus, if CFL's were closed under difference, they would be closed under intersection, but they are not.

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.
  - PDA's accept by final state.

DFA and PDA in Parallel



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 $\delta_A(q,a) = q$.

Formal Construction – (2)

- Final states of combined PDA are those [q,p] such that q is a final state of A and p is an accepting state of P.
- Initial state is the pair ($[q_0,p_0]$ consisting of the initial states of each.
- Easy induction: $([q_0,p_0], w, Z_0)\vdash^*([q,p], \varepsilon, \alpha)$ if and only if $\delta_A(q_0,w) = q$ and in P: $(p_0, w, Z_0)\vdash^*(p, \varepsilon, \alpha)$.