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 more theory (Turing machines, decidability) to prove no algorithm exists.

Testing Emptiness

- Can detect variables that generate no terminal string.
  - Mark all \( V \) s.t. \( V \rightarrow \) terminals
  - Mark \( U \) if \( U \rightarrow \) terminals & marked vars
  - Continue until no more marks possible
- If the start symbol is not marked, then the CFL is empty;

Testing Membership

- Want to know if string \( w \) is in \( L(G) \).
- Assume \( G \) is in Chomsky normal form.
  - 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 \( X \) containing sets of variables.
  - \( X_{ij} = \{ \text{variables } A \mid A \rightarrow * a_i...a_j \} \).
- Induction on \( j-i+1 \).
  - The length of the derived string.
- Finally, ask if \( S \) is in \( X_{1n} \).
**CYK Algorithm – (2)**

- **Basis:** \( X_{ij} = \{ A \mid A \rightarrow a \text{ is a production} \} \).
- **Induction:** \( X_{ij} = \{ A \mid \text{there is a production } A \rightarrow BC \text{ and an integer } k, \text{ with } i < k < j, \text{ such that } B \text{ is in } X_{ik} \text{ and } C \text{ is in } X_{k+1,j} \} \).

**Example: CYK Algorithm**

Grammar: \( S \rightarrow AB, \ A \rightarrow BC \mid a, \ B \rightarrow AC \mid b, \ C \rightarrow a \mid b \)
String \( w = ababa \)

\[
\begin{align*}
X_{11} &= \{ A, C \} \\
X_{22} &= \{ B, C \} \\
X_{33} &= \{ A, C \} \\
X_{44} &= \{ B, C \} \\
X_{55} &= \{ A, C \}
\end{align*}
\]

\[
\begin{align*}
X_{12} &= \{ B, S \} \\
X_{23} &= \{ A \} \\
X_{34} &= \{ B, S \} \\
X_{45} &= \{ A \}
\end{align*}
\]

Yields nothing
Testing Infiniteness

- The idea is like that for regular languages.
- Look for “loop” in CNF grammar

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 $S \rightarrow S_1S_2$.
- 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 $S \rightarrow S_1S \mid \varepsilon$.
- 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 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 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 $h(a)$.

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: $S \rightarrow AB, A \rightarrow 0A1 \mid 01, B \rightarrow 2B \mid 2$.
  - So is $L_3 = \{0^n2^n \mid n \geq 1, i \geq 1\}$.
  - 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

- DFA and PDA run in parallel.
- Accept if both accept.

Formal Construction (Skip)

- Let the DFA A have transition function $\delta_A$.
- Let the PDA P have transition function $\delta_P$.
- 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 $(r, \alpha)$.
  - Note $a$ could be $\epsilon$, in which case $\delta_A(q,a) = q$.

Formal Construction – (2) skip

- 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:** $([q_0,p_0], w, Z_0)^{\ast}$ if and only if $\delta_A(q_0,w) = q$ and in P: $(p_0, w, Z_0)^{\ast} (p, \epsilon, \alpha)$.