CS423 Finite Automata & Theory of Computation

TTh 12:30 - 13:50 in Smal Physics Lab 111 (section 1)

TTh 9:30 - 10:50 in Blow 331 (section 2)

Prof. Weizhen Mao, wxmaox@wm.edu, wm@cs.wm.edu

General Information

- Office Hours: TTh 11:00 - 12:00 in 114 McGl and W 2:30 - 3:00 on zoom or by email
- Grader: Jay Idema for section 1 (zoom office hour https://cwm.zoom.us/j/98189667842?pwd=bEFNa0NsWkh1dlRm
- Grader: Toon Tran for section 2 (zoom office hour https://cwm.zoom.us/j/8420633189)
- Textbook: Intro to the theory of computation (any edition), Michael Sipser. An e-book in PDF maybe available online.
- Prerequisites/background: Linear algebra, Data structures and algorithms, and Discrete math

5.1 Context-free grammars (*Sipser 2.1 pp. 100-105*)

- CFG $G = (V, \Sigma, R, S)$, where V is the set of variables, Σ is the set of terminals (alphabet), R is the set of rules in the form of $V \rightarrow (V \cup \Sigma)^*$ (head$\rightarrow$body), and $S \in V$ is the start variable.
- The CFG that generates all palindromes (strings that read the same forward and backward) over $\{0, 1\}$ is $G = (\{S\}, \{0, 1\}, R, S)$, where R contains $S \rightarrow 0S0|1S1|0|1|\varepsilon$.
- Any language that can be generated by a CFG is called context-free.

- Let u, v, w be strings in $(V \cup \Sigma)^*$. If $A \to w$ is a rule, then uAv yields uwv, written $uAv \Longrightarrow uwv$. We say u derives v, written $u \stackrel{*}{\Rightarrow} v$, if $\exists u_1, \ldots, u_k \in (V \cup \Sigma)^*$ such that $u \Longrightarrow u_1 \Longrightarrow \cdots \Longrightarrow u_k \Longrightarrow v$. Here, $\Longrightarrow$ means one step and $\stackrel{*}{\Rightarrow}$ means zero or more steps.
- Leftmost and rightmost derivations: $\underset{lm}{\Rightarrow}$, $\underset{lm}{\stackrel{*}{\Rightarrow}}$, $\underset{rm}{\Rightarrow}$, $\underset{rm}{\stackrel{*}{\Rightarrow}}$.
- The language of a CFG G, $L(G) = \{w \in \Sigma^* | S \stackrel{*}{\Rightarrow} w\}$. $L(G)$ is said to be a CFL.

Some Simple CFGs and their CFLs

Example 1: $L = \{0^n 1^n | n \geq 0\}$ a context-free language. It can be generated by the following context-free grammar.

$S \rightarrow 0S1|\varepsilon$.

Example 2: Given a CFG G, describe $L(G)$.

$S \rightarrow AA$, $A \rightarrow AAA|bA|Ab|a$
Leftmost derivation: $S \Rightarrow AA \Rightarrow bAA \Rightarrow baA \Rightarrow baa$
Rightmost derivation: $S \Rightarrow AA \Rightarrow Aa \Rightarrow bAa \Rightarrow baa$

$L(G) = \{w \in \{a,b\}^* | w$ has an even (nonzero) number of a's$\}$.

Example 3: A CFG for simple expressions in programming languages:

$S \rightarrow S+S|S*S|(S)|I$
$I \rightarrow Ia|Ib|I0|I1|a|b$

5.2 Parse trees and ambiguity

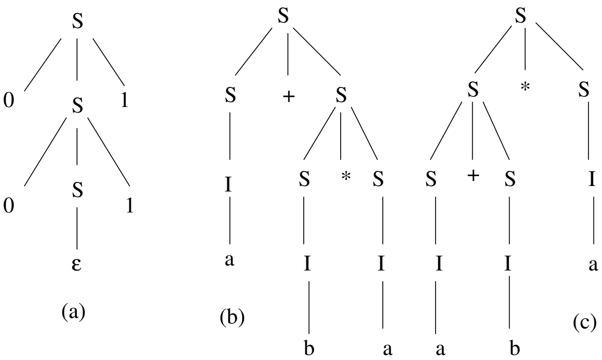

Figure 1: Parse trees and ambiguity

(a) Parse tree and derivation $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0011$
(b) and (c) Two parse trees (or derivations) for string $a + b * a$.

Parse trees

- A parse tree is a tree representation for a derivation, in which each interior node is a variable, each leaf node is either a terminal, or ε, and if an interior node is a variable A and its children are $X_1, \ldots, X_k$, then there must be a rule $A \rightarrow X_1 \cdots X_k$.
- Yield of a parse tree: Concatenation of the leaf nodes in a parse tree rooted at the start variable.
- Four equivalent notions:
  1. $S \overset{*}{\Rightarrow} w$;
  2. $S \underset{lm}{\overset{*}{\Rightarrow}} w$;
  3. $S \underset{rm}{\overset{*}{\Rightarrow}} w$; and
  4. A parse tree with root S and yield w.

Ambiguity in grammars and languages (*Sipser 2.1 pp. 105-106*)

- A CFG $G = (V, \Sigma, R, S)$ is ambiguous if there is $w \in \Sigma^*$ for which there are at least two parse trees (or leftmost derivations).
- Grammar G: $S \rightarrow S + S | S * S | (S) | I$ and $I \rightarrow Ia | Ib | I0 | I1 | a | b$ is ambiguous since $a + b * a$ has two parse trees.
- Some ambiguous grammars have an equivalent unambiguous grammar. For example, an unambiguous grammar for the simple expressions is G': $S \rightarrow S + T | T$, $T \rightarrow T * F | F$, $F \rightarrow (S) | I$, and $I \rightarrow Ia | Ib | I0 | I1 | a | b$.

- A context-free language is said to be inherently ambiguous if all its grammars are ambiguous.
- There is no algorithm to determine whether a given CFG is ambiguous. There is no algorithm to remove ambiguity from an ambiguous CFG. There is no algorithm to determine whether a given CFL is inherently ambiguous.

Chomsky normal form (*Sipser 2.1 pp. 106-109*)

The Chomsky Normal Form (CNF): Any nonempty CFL without ε has a CFG G in which all rules are in one of the following two forms: $A \to BC$ and $A \to a$, where A, B, C are variables, and a is a terminal. Note that one of the uses of CNF is to turn parse trees into binary trees.

5.3 More CFGs design

Example 1: $\{a^m b^n c^{m+n} | m, n \geq 0\}$

- Rewrite the pattern as $a^m b^n c^n c^m$
- $S \to aSc | T$, $T \to bTc | \varepsilon$

Example 2: $\{\underline{a^m b^m}\ \underline{c^n d^n} | m, n \geq 0\} \cup \{a^m \underline{b^n c^n} d^m | m, n \geq 0\}$

- $S \to S_1 | S_2$
- $S_1 \to AB$, $A \to aAb | \varepsilon$, $B \to cBd | \varepsilon$
- $S_2 \to aS_2d | C$, $C \to bCc | \varepsilon$

Example 3: $\{0^m1^n | m \neq n\}$

- Rewrite the language as $\{0^m1^n | m < n\} \cup \{0^m1^n | m > n\}$, which is $\{0^m1^{n-m}1^m\} \cup \{0^n0^{m-n}1^n\}$
- $S \rightarrow S_1 | S_2$
- $S_1 \rightarrow 0S_11 | A$, $A \rightarrow 1A | 1$
- $S_2 \rightarrow 0S_21 | B$, $B \rightarrow 0B | 0$

Example 4: Given the following grammar CFG G, what is its language?

- $S \rightarrow aS | Sb | a | b$
- $S \rightarrow aSbS | bSaS | \varepsilon$

Example 5: $L = \{a^i b^j c^k \in \{a,b,c\}^* \mid i+j \neq k\}$
The grammar will pair a and c until running out one of the two.
Then the grammar will consider the following cases.
$L = L_1 \cup L_2 \cup L_3$

- Case 1: $i = k$. Then $j \neq 0$, i.e., $j \geq 1$. So $L_1 = \{a^i b^+ c^i\}$
- Case 2: $i > k$. Then $j \geq 0$. So $L_2 = \{a^k a^+ b^* c^k\}$
- Case 3: $i < k$. And $j \neq k - i$. So $L_3 = \{a^i b^j c^{k-i} c^i\}$

$S \to aSc \mid S_1 \mid S_2 \mid S_3$
$S_1 \to bS_1 \mid b$ (Case1: To generate b^+)
$S_2 \to aS_2 \mid S_2 b \mid a$ (Case 2: To generate $a^+ b^*$)
$S_3 \to bS_3 c \mid S_1 \mid C$, $C \to cC \mid c$ (Case 3: To generate b^j and c^{k-i} s.t. $j \neq k - i$)

Example 6: $L = \{a^i b^j \mid i \neq j \text{ and } 2i \neq j\}$.
Draw an x-axis and mark two points, one for i and one for $2i$. These two points divides the x-axis into three intervals: $j < i$, $i < j < 2i$, and $j > 2i$.

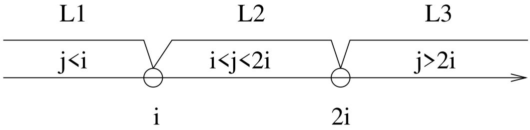

Figure 2: Intervals that j falls in

$L = L_1 \cup L_2 \cup L_3$: $S \to S_1 \mid S_2 \mid S_3$
$L_1 = \{a^i b^j \mid j < i\}$: $S_1 \to aS_1 b \mid A$, $A \to aA \mid a$
$L_2 = \{a^i b^j \mid i < j < 2i\}$: $S_2 \to aS_2 b \mid aTb$, $T \to aTbb \mid abb$
$L_3 = \{a^i b^j \mid j > 2i\}$: $S_3 \to aS_3 bb \mid B$, $B \to Bb \mid b$

Example 7: $L = \{x\#y \mid x, y \in \{0,1\}^*, |x| \neq |y|\}$
Consider two cases: $|x| < |y|$ and $|x| > |y|$.

- Case 1 $|x| < |y|$: $x = 01$ and $y = 100$. String $x\#y = 01\#100$. After pairing 01 and 00, what's left is $\#1$.
- Case 2 $|x| > |y|$: $x = 110$ and $y = 00$. String $x\#y = 110\#00$. After pairing 11 and 00, what's left is $0\#$.

$S \to 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid T\# \mid \#T$
$T \to 0T \mid 1T \mid 0 \mid 1$

6.1 PDAs (*Sipser 2.2 pp. 102-114*)

- PDA = NFA + Stack (still with limited memory but more than that in FAs)
- PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where
  - Q: A finite set of states
  - Σ: A finite set of input symbols (input alphabet)
  - Γ: A finite set of stack symbols (stack alphabet)
  - δ: The transition function from $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})$ to $2^{Q \times (\Gamma \cup \{\varepsilon\})}$
  - q_0: The start state
  - F: The set of final states

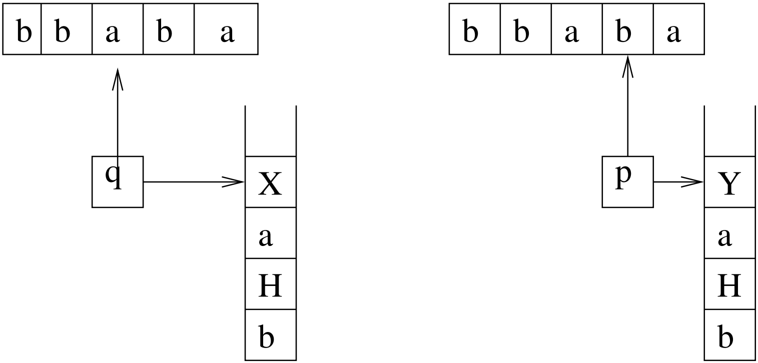

Figure 3: How a transition step occurs within a PDA if $\delta(q, a, X) = \{(p, Y)\}$

- What does $\delta(q, a, X) = \{(p, Y)\}$ mean? If the current state is q, the current input symbol is a, and the stack symbol at the top of the stack is X, then the automaton changes to state p and replace X by Y.
- What if ε replaces a, or X, or Y? For example,
  $\delta(q, \varepsilon, X) = \{(p, Y)\}$: No cursor move. X replaced by Y (pop + push)
  $\delta(q, a, \varepsilon) = \{(p, Y)\}$: Push Y
  $\delta(q, a, X) = \{(p, \varepsilon)\}$: Pop X
  $\delta(q, \varepsilon, \varepsilon) = \{(p, Y)\}$: No cursor move. Push Y
  $\delta(q, a, \varepsilon) = \{(p, \varepsilon)\}$: No stack change
  $\delta(q, \varepsilon, X) = \{p, \varepsilon)\}$: No cursor move. Pop X
  $\delta(q, \varepsilon, \varepsilon) = \{(p, \varepsilon)\}$: No change except state
- The state diagram of PDAs: For transition $\delta(q, a, X) = \{(p, Y)\}$, draw an arc from state q to state p labeled with $a, X \to Y$.

- Instantaneous description (ID) of a PDA: (q, w, γ) represents the configuration of a PDA in the state of q with the remaining input of w yet to be read and the stack content of γ. (The convention is that the leftmost symbol in γ is at the top of the stack.)
- Binary relation $\vdash$ on ID's: $(q, aw, X\beta) \vdash (p, w, Y\beta)$ if $\delta(q, a, X)$ contains (p, Y). $\vdash$ represents one move of the PDA, and $\overset{*}{\vdash}$ represents zero or more moves of the PDA.
- Language of a PDA M (or language recognized by M) is $L(M) = \{w | (q_0, w, \varepsilon) \overset{*}{\vdash} (f, \varepsilon, \gamma) \text{ for } f \in F\}$.
- How does a PDA check the stack is empty? At the beginning of any computation, it pushes a special symbol \$ to the initially empty stack by having transition $\delta(q_0, \varepsilon, \varepsilon) = \{(q, \$)\}$.

Example 1 (Sipser p. 112): A PDA that recognizes $\{0^n 1^n | n \geq 0\}$.

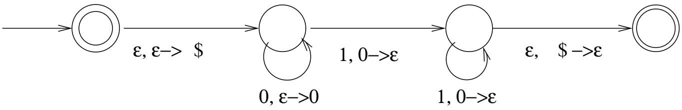

Figure 4: An example of a PDA

Example 2 (Sipser p. 114): A PDA that recognizes $\{a^i b^j c^k | i, j, k \geq 0, i = j \text{ or } i = k\} = \{a^n b^n c^*\} \cup \{a^n b^* c^n\}$.

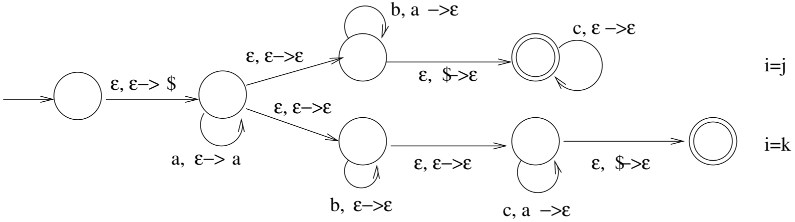

Figure 5: Another PDA

Example: How the PDA in the above example accepts input *aabbcc*.

Theorem: The equivalence of PDA, CFG, and CFL

7.1 Proving non-CFLs by pumping lemma (*Sipser 2.3 (pp. 125-129)*)

Theorem 2.34 (The pumping lemma for CFLs)
Let A be a CFL. Then there exists a constant p such that $\forall s \in A$ with $|s| \geq p$, we can write $s = uvxyz$ such that

1. $|vy| > 0$; (not allow $v = y = \varepsilon$)
2. $|vxy| \leq p$; and
3. $\forall i \geq 0$, string $uv^i xy^i z \in A$.

Recall in the PL for RLs, s is partitioned into x, y, z satisfying

1. $|y| > 0$;
2. $|xy| \leq p$; and
3. $\forall i \geq 0$, string $xy^i z \in A$

How to use the pumping lemma to prove that a language A is not Context-free?

- Assume that A is context-free by contradiction. Then the pumping lemma applies to A. Let p be the constant in the pumping lemma. (Always begin the proof with these three steps.)
- Select $s \in A$ with $|s| = f(p) \geq p$. (This may be tricky. But start with an intuitive simple string that uses p in the length.)
- By the pumping lemma, $s = uvxyz$ with (1)$|vy| > 0$, (2)$|vxy| \leq p$, and (3)$uv^ixy^iz \in A$, $\forall i \geq 0$.
- Prove for ANY u, v, x, y, z such that $s = uvxyz$, $|vy| > 0$, and $|vxy| \leq p$, find $i \geq 0$ such that $uv^ixy^iz \notin A$.
  A contradiction to (3) in PL!

Example 2.36 (Sipser p.128): Prove $B = \{a^n b^n c^n | n \geq 0\}$ is non-CF.

Pf. Assume B is CF. Then PL applies. Let p be the constant.

- Select $s = a^p b^p c^p = a \cdots b \cdots c \cdots \in B$, with $|s| = 3p > p$.
- By PL, $s = uvxyz$ with $|vy| > 0$ and $|vxy| \leq p$.
- Consider what vxy can be. Imagine vxy is a sliding window that moves left to right within s.
- Case 1: vxy contains one symbol type (a^+, b^+, or c^+)
- Case 2: vxy contains two symbol types ($a^+ b^+$ or $b^+ c^+$)
- Choose $i = 0$ for both cases. Then $uv^0 xy^0 z \notin B$ for both.
- A contradiction to the PL!

Example 2.38 (Sipser p. 129): Prove $D = \{ww | w \in \{0,1\}^*\}$ is non-CF.

Pf. Three sentences to start the proof. Then,

- Select $s = 0^p 1 0^p 1 = 0 \cdots 010 \cdots 01 \in D$, $|s| = 2p + 2 > p$.
- By PL, $s = uvxyz$ with $|vy| > 0$ and $|vxy| \leq p$
- We consider the following partition: $s = 0^{p-1} \cdot 0 \cdot 1 \cdot 0 \cdot 0^{p-1} 1$, where $u = 0^{p-1}$, $v = 0$, $x = 1$, $y = 0$, and $z = 0^{p-1} 1$.
- For any $i \geq 0$, $uv^i xy^i z = 0^{p-1} \cdot 0^i \cdot 1 \cdot 0^i \cdot 0^{p-1} 1 = 0^{p-1+i} \cdot 1 \cdot 0^{i+p-1} \cdot 1 \in D$
- No contradiction. Need to choose a different s.
- Try $s = 0^p 1^p 0^p 1^p$ and $i = 0$ (Exercise or read p. 129 bottom)

Example 3: Prove that $A = \{0^j 1^{j^2}\}$ is non-CF.

- Select $s = 0^p 1^{p^2} = uvxyz \in A$, where $|vy| > 0$ and $|vxy| \leq p$.
- $vxy = 0^+, 1^+, \text{or } 0^+1^+$ (three cases)
- For the first two cases, choose $i = 0$ to shrink the 0 and 1 blocks, respectively, thus making the string $uxz \notin A$
- Case 3: $vxy = 0^+1^+$.
- Case 3.1: v or y contains both 0 and 1, i.e., $v \text{ or } y = 0^+1^+$. Let $i = 2$. Then uv^2xy^2z contains substring $0^+1^+0^+1^+$, thus not in A.
- Case 3.2: v and y do not contain both 0 and 1, which means that v and y each contain at most one symbol type, i.e. $v = 0^{|v|}$ and $y = 1^{|y|}$. (Note: v or y may be ε but not both)

Continue with Case 3.2

- Let $i = 2$. Then $uv^2xy^2z = 0^{p+|v|} \cdot 1^{p^2+|y|}$
- Is $(p+|v|)^2 = (p^2+|y|)$?
- Is $p^2+2p|v|+|v|^2 = p^2+|y|$?
- Is $2p|v|+|v|^2 = |y|$? Or Left = Right?
- If $|y| = 0$ and $|v| \neq 0$: Left $>$ Right
- If $|y| \neq 0$ and $|v| = 0$: Left $<$ Right
- If $|y| \neq 0$ and $|v| \neq 0$:
  Left $= 2p|v|+|v|^2 > p \geq |vxy| \geq |y| =$ Right
  So Left $>$ Right
- So for all combinations of $|v|$ and $|y|$, Left $\neq$ Right. So $uv^2xy^2z \notin A$
- A contradiction!

Example 4: Prove that $L = \{a^i b^j c^i d^j \mid i, j \geq 0\}$ is non-CF.

- Assume L is CF. Then the PL applies to L. Let p be the constant.
- Select $s = a^p b^p c^p d^p$. $s \in L$ and $|s| = 4p > p$.
- By PL, $s = uvxyz$ with $|vy| > 0$ and $|vxy| \leq p$.
- Since $|vxy| \leq p$, v and y cannot contain both a's and c's, nor can it contain both b's and d's. Further $|vy| > 0$. We have $uv^0xy^0z = uxz \notin L$, because it either contains fewer a's than c's, or fewer c's then a's, or fewer b's than d's, or fewer d's than b's.
- A contradiction to the PL.
- So L is non-CF.

7.2 Proving non-CFLs by closure properties

- Closed under union: If A and B are CF, so is $A \cup B$.
  Proof: CFG G_A with $S_A \to \cdots$ and CFG G_B with $S_B \to \cdots$.
  Define a CFG that generates $A \cup B$ as $S \to S_A \mid S_B$ plus the grammars G_A and G_B.
- Closed under concatenation: If A and B are context-free, so is AB.
  Proof: CFG G_A with $S_A \to \cdots$ and CFG G_B with $S_B \to \cdots$.
  Define a CFG that generates AB as $S \to S_A S_B$ plus the grammars G_A and G_B.
- Closed under star: If A is context-free, so is A^*.
  Proof: Consider CFG G with $S_1 \to \cdots$. Define a CFG that generates A^* as $S \to SS_1 \mid \varepsilon$ plus the grammar G.

- Closed under reverse: If A is context-free, so is A^R.
- Not closed under intersection: Consider $A = \{a^n b^n c^m\}$ and $B = \{a^m b^n c^n\}$.
- Not closed under complementation: Note that $A \cap B = \overline{\overline{A} \cup \overline{B}}$.
- Not closed under difference: Note that $\overline{A} = \Sigma^* - A$.

- Intersect with a regular language: If A is context-free and B is regular, then $A \cap B$ is context-free.
- Difference from a regular language: If A is context-free and B is regular, then $A - B$ is context-free. Note that $A - B = A \cap \overline{B}$.

How can closure properties of CFLs be used to prove that a given language is non-CF?

Example 1: $A = \{w \in \{a, b, c\}^* | n_a(w) = n_b(w) = n_c(w)\}$ is non-CF. (Note: $n_a(w)$ is defined to be the number of a's in w.)

- Assume A is CF. Let $B = \{a^* b^* c^*\}$, a RL.
- $A \cap B = \{a^n b^n c^n\}$ must be CF by the closure property of the intersection of a CFL and a RL.
- But we proved before that $\{a^n b^n c^n\}$ is non-CF.
- A contradiction.
- So A must be non-CF.

How can closure properties of CFLs be used to prove that a given language is CF?

Example 2: $B = \{a^i b^j c^k | j > i + k\}$ is CF.

- Since $j > i + k$, let $j = i + k + h$ for some $h > 0$.
- $a^i b^j c^k = a^i \cdot b^{i+k+h} \cdot c^k = a^i \cdot b^{i+h} \cdot b^k \cdot c^k = (a^i b^i)(b^+)(b^k c^k)$
- This is the concatenation of three CFLs.
- By the closure property of CFLs under concatenation, B is CF.