Computational Models - Lecture 6[1]

Handout Mode

Nachum Dershowitz & Yishay Mansour .

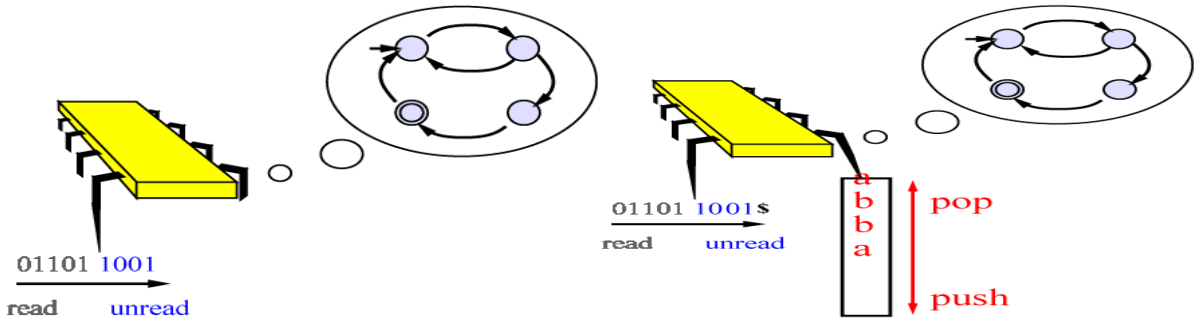
Tel Aviv University.

May 3–8, 2017

[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Outline

- Pumping Lemma for context free languages
- Closure properties for CFL and testing properties.



- Sipser's book, 2.1, 2.2&2.3

- **Midterm: May 26th, 2017 !!!**

CFGs and CFLs, reminder

A context-free grammar is a 4-tuple (V, Σ, R, S), where

- V is a finite set of variables
- Σ is a finite set of terminals $(V \cap \Sigma = \emptyset)$
- R is a finite set of rules of the form $A \to x$, where $A \in V$ and $x \in (V \cup \Sigma)^*$.
- $S \in V$ is the start symbol.

- Let $u, v \in (V \cup \Sigma)^*$. If $A \to w \in R$, then uAv yields uwv, denoted $uAv \to uwv$.
- $u \stackrel{*}{\to} v$ if $u = v$, or $u \to u_1 \to \ldots \to u_k \to v$ for some sequence $u_1, u_2, \ldots, u_k$

Definition 1

The language of the grammar G, denoted $\mathcal{L}(G)$, is $\{w \in \Sigma^* : S \stackrel{*}{\to} w\}$

where $\stackrel{*}{\to}$ is determined by G.

Part I

Non-Context-Free Languages

Proving a Language is not a CFL

- The pumping lemma for finite automata and Myhill-Nerode theorem are our tools for showing that languages are not regular.
- We will now show a similar pumping lemma for context-free languages.
- It is slightly more complicated . . .

Pumping Lemma for CFL (also known as, the *uvxyz* Theorem)

Theorem 2

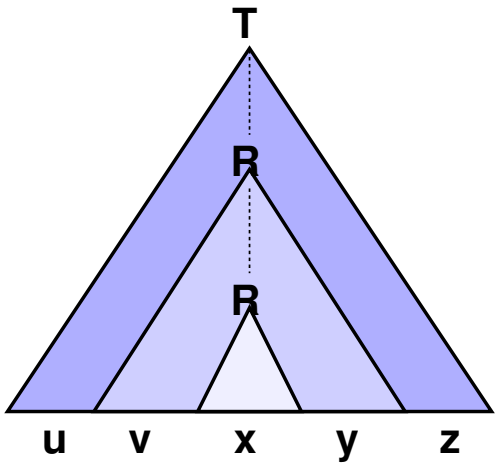
For any CFL $\mathcal{L}$ there exists $\ell \in \mathbb{N}$ ("critical length"), such that for any $w \in \mathcal{L}$ with $|w| \geq \ell$, there exit $u, v, x, y, z \in \Sigma^*$ such that $w = uvxyz$ and

- *For every $i \geq 0$: $uv^i xy^i z \in \mathcal{L}$*
- *$|vy| > 0$, ("non-triviality")*
- *$|vxy| \leq \ell$*

Basic Intuition:
Let $\mathcal{L}$ be a CFL and a let w be a "very long" string in $\mathcal{L}$. Then w must have a "tall" parse tree.

Hence, some root-to-leaf path must repeat a symbol. Why is that so?

Basic Intuition cont.



We have: $T \stackrel{*}{\rightarrow} uRz$, $R \stackrel{*}{\rightarrow} vRy$, and $R \stackrel{*}{\rightarrow} x$.

Proof of Thm 2

Let G be a CFG and let $\mathcal{L} = \mathcal{L}(G)$.

- Let b be the max number of symbols in right-hand-side of any rule (what is b for a CNF grammar?).

  Since no node in a parse tree of G has more than b children, at depth d such tree has at most b^d leaves.

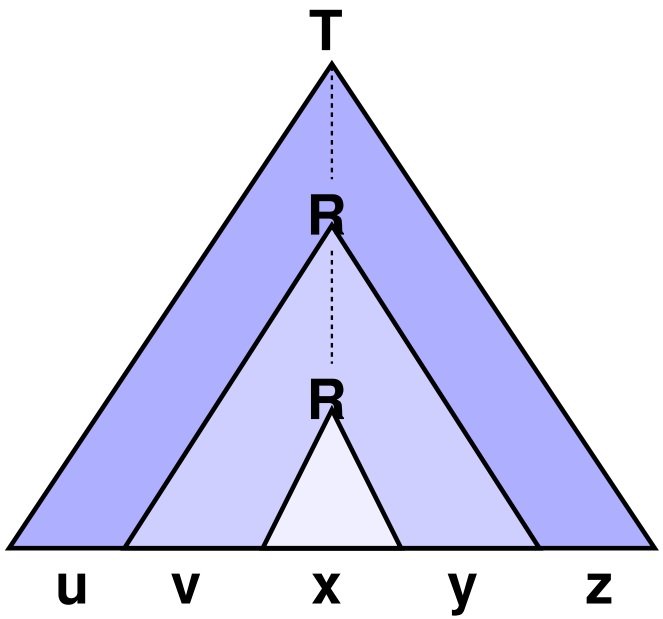
- Let $|V|$ be the number of variables in G, and set $\ell = b^{|V|+2}$.

Let w be a string with $|w| \geq \ell$, and let T be parse tree for w (with respect to G) with fewest nodes

- T has height $\geq |V| + 2$
- Some path in T has length $\geq |V| + 2$
- Such path repeats a variable R

Proof of Thm 2 cont.

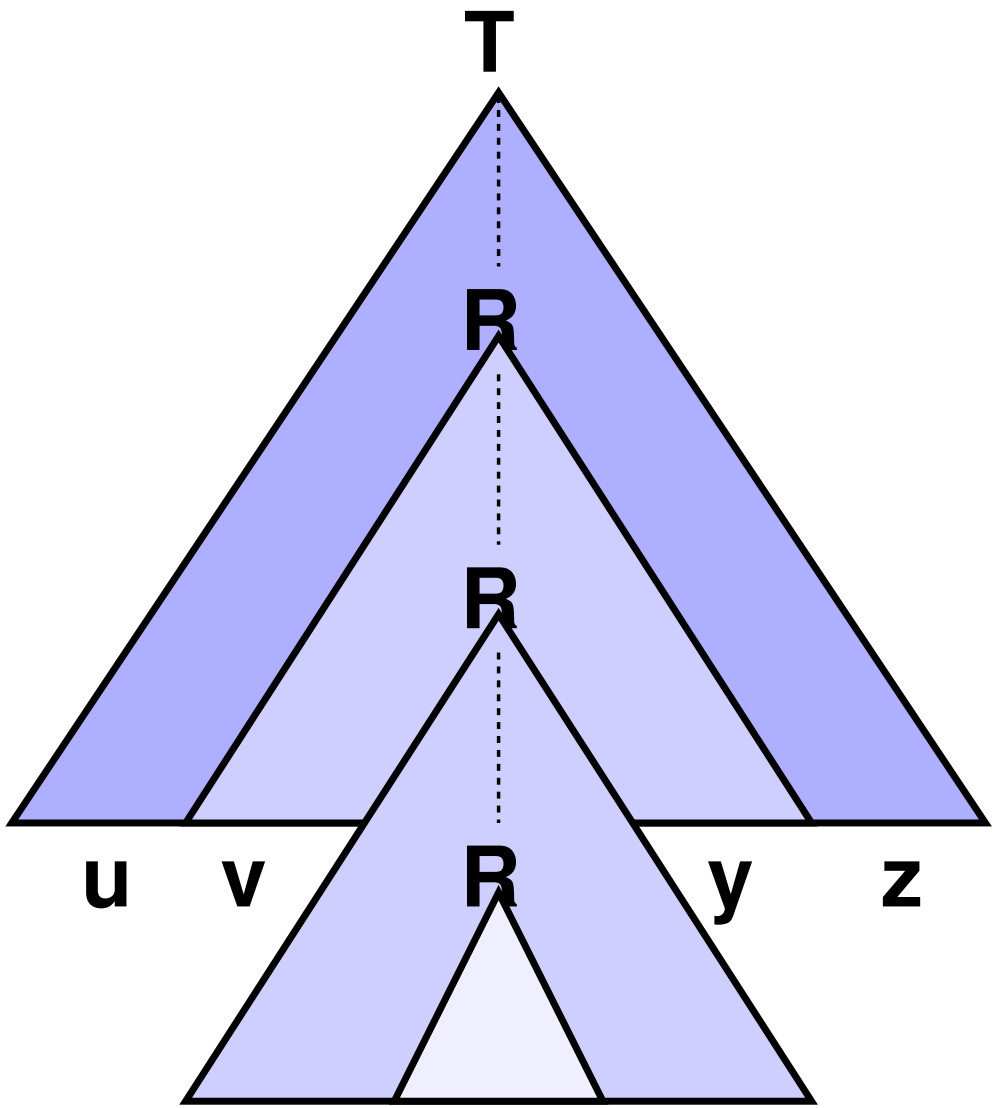
Set $w = uvxyz$



- Each occurrence of R produces a string
- Upper produces string vxy
- Lower produces string x

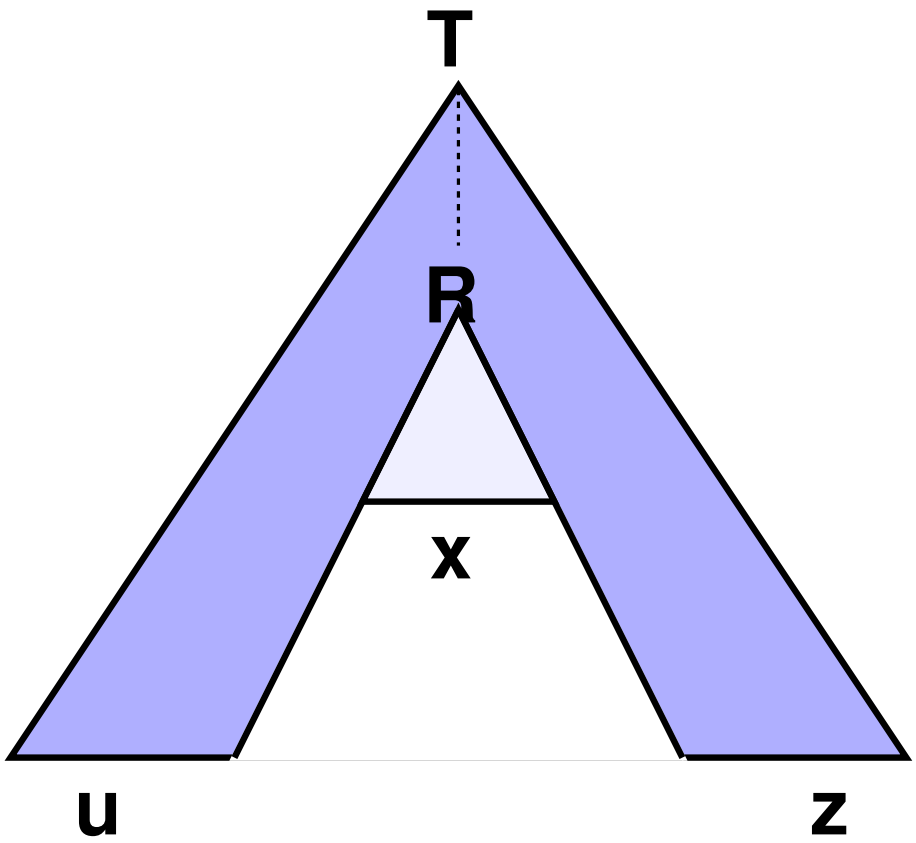
Proving $uv^ixy^iz \in \mathcal{L}$ for all $i > 1$

Replacing smaller by larger yields uv^ixy^iz, for $i > 0$.



Proving $uv^i xy^i z \in \mathcal{L}$ for $i = 0$

Replacing larger by smaller yields uxz.

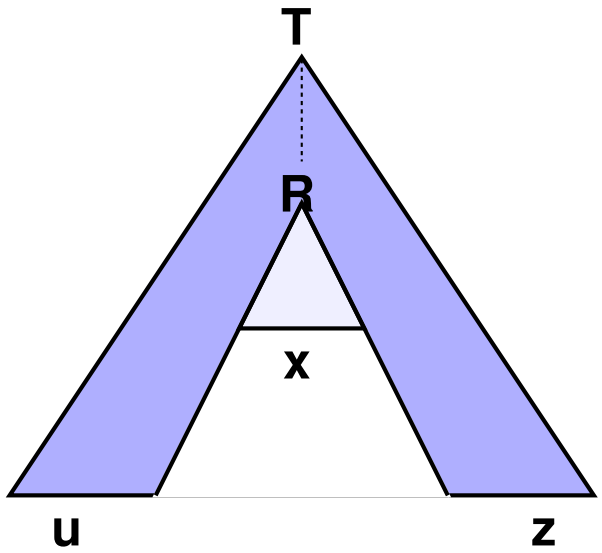


Together, they establish:

- $uv^i xy^i z \in \mathcal{L}$ for all $i \geq 0$

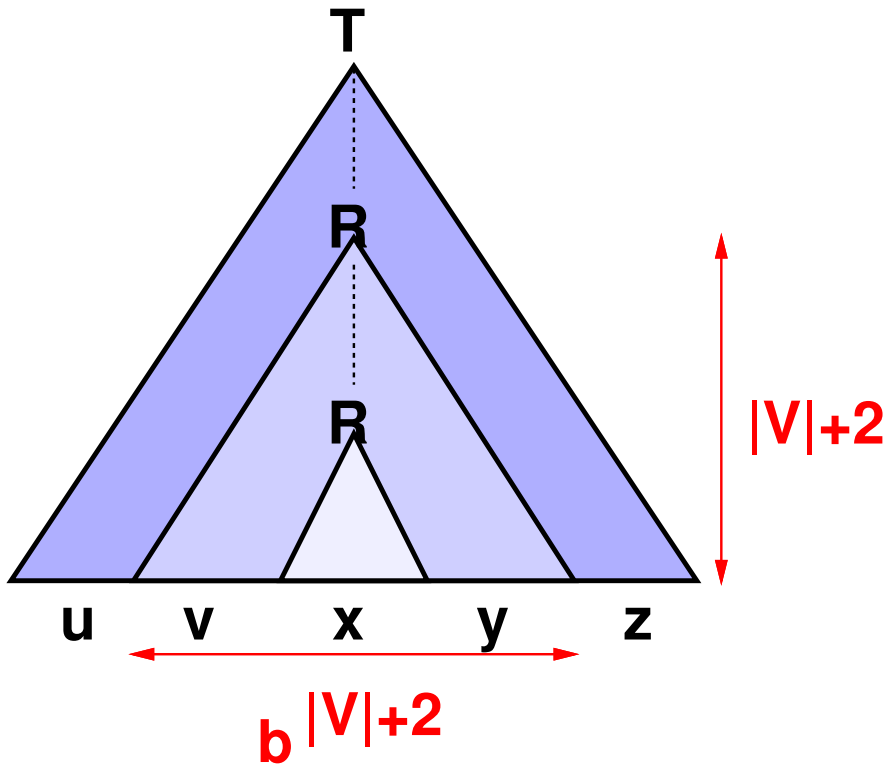
Proving $|vy| > 0$

If v and y are both ε, then



is a parse tree for w with fewer nodes than T, a contradiction.

Proving $|vxy| \leq \ell$



- Without loss of generality both occurrences of R lie in bottom $|V| + 1$ variables on the path.
- The upper occurrence of R (from now on R^1) generates vxy.
- Subtree rooted at R^1 is of height at most $|V| + 2$.

  Hence, $|vxy| \leq b^{|V|+2} = \ell$.

Non CFL Example (1)

Claim 3

$\mathcal{L}_1 = \{a^n b^n c^n : n \in \mathbb{N}\}$ is not a CFL.

Proof: By contradiction. Assume $\mathcal{L}_1$ is a CFL with grammar G, let ℓ be the critical length of G and consider $w = a^\ell b^\ell c^\ell$. Let u, v, x, y, z be the strings with $w = uvxyz$ guaranteed by Thm 2 for w.

- Both v and y are monochromatic — repetitions of of a single letter.

  Proof: Otherwise, uv^2xy^2z would have out-of-order symbols.

- Hence, uv^2xy^2z is imbalanced

♣

Non CFL Example (2)

Claim 4

$\mathcal{L}_2 = \{a^i b^j c^k : 0 \leq i \leq j \leq k\}$ is not context free.

Proof: By contradiction. Assume $\mathcal{L}_2$ is a CFL with grammar G, let ℓ be the critical length of G and consider $w = a^\ell b^\ell c^\ell$. Let u, v, x, y, z be the strings with $w = uvxyz$ guaranteed by Thm 2 for w.

Both v and y are monochromatic.

- If neither v nor y contains a, then uv^0xy^0z has too many a's.
- If neither v nor y contains c, then uv^2xy^2z has too few c's.
- If neither v nor y contains b, then
  - either $|v| > 0$, and then uv^2xy^2z has more a's than b's.
  - or $|y| > 0$, and then uv^0xy^0z has more b's than c's

♣

Non CFL Example (3)

Claim 5

$\mathcal{L}_3 = \{ww : w \in \{0,1\}^*\}$ is not context-free.

Proof:
By contradiction. Assume $\mathcal{L}_3$ is a CFL with grammar G, let ℓ be the critical length of G and consider $w = 0^\ell 1^\ell 0^\ell 1^\ell$. Let u, v, x, y, z be the strings with $w = uvxyz$ guaranteed by Thm 2 for w.

- Assuming vxy is in the first half of w, then uv^2xy^2z "moves" a 1 into the first position of second half.
- Assuming vxy is in the second half, then uv^2xy^2z "moves" a 0 into the last position of first half.
- Assuming vxy straddles the midpoint, then pumping *down* to uxz yields $0^\ell 1^i 0^j 1^\ell$ where i and j cannot both be ℓ.

♣

Note that $\{ww^R : w \in \{0,1\}^*\}$ is a CFL.

Part II

Closure Properties of Context-Free Languages

Simple closure properties of CFL's

- CFL's are closed under
  - Union: $S \to S_1 \mid S_2$
  - Concatenation: $S \to S_1 S_2$
  - Star: $S_{new} \to \varepsilon \mid S_{old} \mid S_{old} S_{new}$
- What about complement and intersection?

Intersection

$$S_1 \to A_1 B_1 \qquad S_2 \to A_2 B_2$$
$$A_1 \to 0A_1 1 | \varepsilon \qquad A_2 \to 0A_2 | \varepsilon$$
$$B_1 \to 2B_1 | \varepsilon \qquad B_2 \to 1B_2 2 | \varepsilon$$

$$\mathcal{L}_1 = 0^n 1^n 2^* \qquad \mathcal{L}_2 = 0^* 1^n 2^n$$

- $\mathcal{L}_1 \cap \mathcal{L}_2 = 0^n 1^n 2^n$
- $\mathcal{L}_1$ and $\mathcal{L}_2$ are CFLs (why?),
- But $\mathcal{L}_1 \cap \mathcal{L}_2$ is not a CFL.
- But can't we run two PDA's in parallel, and accept iff both accept??
- What about intersection of a CFL with a regular language?

When CFL Intersects Regular Language

- Are the context free languages closed under intersection with a regular language?
- That is, if $\mathcal{L}_1$ is context free languages, and $\mathcal{L}_2$ is regular, must $\mathcal{L}_1 \cap \mathcal{L}_2$ be context free languages?
- YES!
  - Run PDA $\mathcal{L}_1$ and DFA $\mathcal{L}_2$ "in parallel" (just like the intersection of two regular languages).
  - Formal details omitted (but you should be able to figure them out).

Applications

- Is $\mathcal{L} = \{w \in \{0,1,2\}^* : \#_0(w) = \#_1(w) = \#_2(w)\}$ context free?
  - $\mathcal{L} \cap 0^*1^*2^* = \{0^n1^n2^n : n \geq 0\}$ is not context free.
  - $0^*1^*2^*$ is regular.
  - Context free languages intersected with a regular languages are context free.
  - So $\mathcal{L}$ is not a context free language
- This could also be established using pumping lemma, but proof above is more elegant.

Complementation

The fact that CFLs are not closed under intersection, but are closed under union, implies they are not closed under complementation, as $\mathcal{L}_1 \cap \mathcal{L}_2 = \overline{\overline{\mathcal{L}_1} \cup \overline{\mathcal{L}_2}}$.

We give a simple example where $\mathcal{L}$ is not CFL but $\overline{\mathcal{L}}$ is.

- Take $\mathcal{L} = \{ww : \ w \in \{0,1\}^*\}$.
- $\mathcal{L}$ is not a CFL (why?)
- We prove that $\overline{\mathcal{L}}$ is a CFL

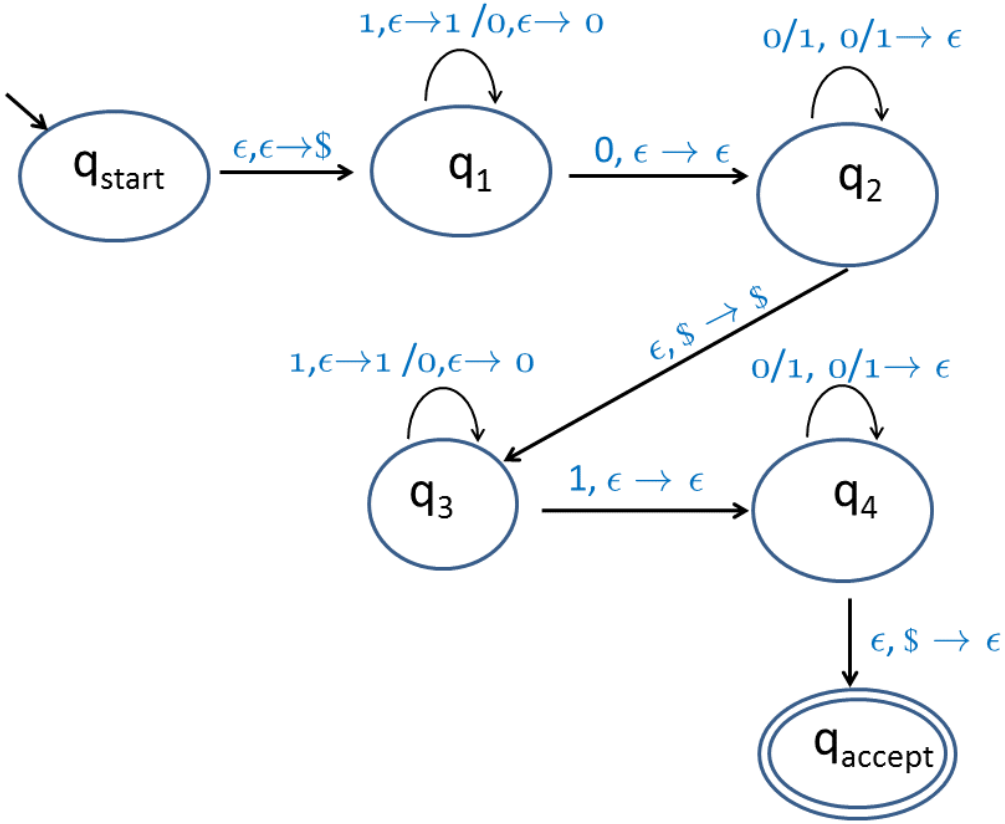
Complementation cont.

- For any $y \in \overline{\mathcal{L}}$, either
  - y's length is odd, or
  - y's length is even, 2ℓ, and $\exists i \geq 1$ such that $y_i \neq y_{\ell+i}$.
- It suffice to construct a PDA/CFG for $\overline{\mathcal{L}}_{even}$ – the even length members of $\overline{\mathcal{L}}$ (why?)
- Let $\overline{\mathcal{L}}^{\sigma}_{even} = \{\{0,1\}^k \sigma \{0,1\}^j \{0,1\}^k \overline{\sigma} \{0,1\}^j : k, j \geq 0\}$
- Note that $\overline{\mathcal{L}}_{even} = \overline{\mathcal{L}}^{0}_{even} \cup \overline{\mathcal{L}}^{1}_{even}$
- and that $\overline{\mathcal{L}}^{\sigma}_{even} = \{\{0,1\}^k \sigma \{0,1\}^k \{0,1\}^j \overline{\sigma} \{0,1\}^j : k, j \geq 0\}$
- CFG for $\overline{\mathcal{L}}^{0}_{even}$

  $S \to AB$
  $A \to CAC \mid 0$
  $B \to CBC \mid 1$
  $C \to 0 \mid 1$

A PDA for $\overline{\mathcal{L}}^0_{even}$

Idea: Guess $k, j \geq 0$, and accept w if it is of the form:
$\{0,1\}^k 0 \{0,1\}^k \{0,1\}^j 1 \{0,1\}^j$



A simpler PDA:

- Loops from q_1 and q_3 changed to $0/1, \varepsilon \to 2$
- Loops from q_2 and q_4 changed to $0/1, 2 \to \varepsilon$

Homomorphism and inverse homomorphism

- *Homomorphism:* replaces each letter with a word
- Example: $h(1) = aba$, $h(0) = aa$
  $h(010) = aa\,aba\,aa$
  $\mathcal{L}_1 = \{0^n 1^n | n \geq 0\}$, $h(\mathcal{L}_1) = \{a^{2n}(aba)^n | n \geq 0\}$.
- Claim: Assuming that $\mathcal{L}$ is a CFL, then so is $h(\mathcal{L})$
- Proof? use the grammar
- *Inverse homomorphism:* $h^{-1}(w) = \{x : h(x) = w\}$,
  $h^{-1}(\mathcal{L}) = \{x : h(x) \in \mathcal{L}\}$
- Example: $\mathcal{L}_2 = \{a^n b^n a^i | n, i \geq 0\}$, $h^{-1}(\mathcal{L}_2) = \{10^i, 0^i | i \geq 0\}$.
- Claim: Assuming that $\mathcal{L}$ is a CFL, then so is $h^{-1}(\mathcal{L})$
- Proof? use the automata (see next)

CFL's are closed under inverse homomorphism

- Idea: let P be a PDA for $\mathcal{L}$. On input word w, emulate $P(h(w))$
- But we cannot afford to store $h(w)$!
- Solution, compute $h(w)$ "on demand":

Algorithm 6

Input w:

1. Initialized a "buffer" Buff to $h(a)$, where a is the first letter of w
2. Emulate a running of P with Buff as its input string.

   Each time Buff is fully read by P, set Buff $= h(a)$, where a is the next letter in w (if exists)
3. `Accept` iff P does

- How do we implement Buff?

CFL's are closed under inverse homomorphism, cont.

Given PDA $(Q, \Sigma, \Gamma, \delta, q_0, F)$ and homomorphism $h\colon \Delta \to \Sigma^*$, we define a new PDA $(Q', \Delta, \Gamma, \delta', q_0', F')$.

- Let $k = \max_{a \in \Delta} |h(a)|$ and $\widetilde{\Sigma} = \cup_{0 \le i \le k} \Sigma^i$.
- $Q' = Q \times \widetilde{\Sigma}$.
- $q_0' = [q_0, \varepsilon]$.
- $F' = F \times \{\varepsilon\}$.
- δ' is define as follows:
  - if $(p, \gamma) \in \delta(q, \varepsilon, \xi)$, then $([p, x], \gamma) \in \delta'([q, x], \varepsilon, \xi)$, for any $x \in \widetilde{\Sigma}$.
  - if $(p, \gamma) \in \delta(q, a, \xi)$, then $([p, x], \gamma) \in \delta'([q, ax], \varepsilon, \xi)$, for any $x \in \widetilde{\Sigma}$.
  - $([q, h(a)], \xi) \in \delta'([q, \varepsilon], a, \xi)$, for any $a \in \Delta$ and $\xi \in \Gamma$.

Part III

Algorithmic Questions

Emptiness of CFGs

Question 7

Given a CFG, G, is $\mathcal{L}(G) = \emptyset$?

In other words, is there a string generated by G?

Theorem 8

There is an algorithm that solves this problem (and always halts).

Possible approaches for a proof:

- Bad Idea: We know how to test whether $w \in \mathcal{L}(G)$ for any string w, so just try it for each w...
- Better Idea: Can the start variable generate a string of terminals?
- A more holistic approach: Can a particular variable generate a string of terminals?

Checking emptiness

Idea: Mark variables that can produce a string of terminals

Algorithm 9 (Deciding $\mathcal{L}(G) = \emptyset$)

1. Mark all terminal symbols in G.
2. Repeat until no new variable become marked:

   Mark any A where $A \to U_1 U_2 \dots U_k$ and all U_i have already been marked.
3. Remove all unmarked variables, and any rule they appear in.
4. If S is removed, then $\mathcal{L}(G) = \emptyset$.

Correctness?

CFGs fullness

Question 10

Given a CFG G, is $\mathcal{L}(G) = \Sigma^*$?

- We just saw an algorithm to determine, given a CFG G, whether $\mathcal{L}(G) = \emptyset$
- $\mathcal{L}(G) = \Sigma^*$ iff $\overline{\mathcal{L}(G)} = \emptyset$. Why not modify the algorithm so it determines emptiness of the complement?
- Unfortunately, CFGs are not closed under complement.

Fact 11

There is no algorithm to solve CFG fullness.

- We are not prepared to prove this remarkable fact (yet).

Finiteness of CFGs

Question 12

Given a CFG G, is $|\mathcal{L}(G)|$ finite?

First, a useful subroutine.

Algorithm 13 (Removing redundant variables and terminals)

1. Mark all terminal symbols in G.
2. Repeat until no new variable become marked:

   Mark any A where $A \to U_1 U_2 \dots U_k$ and all U_i have already been marked.
3. Remove all unmarked variables, and any rule they appear in.
4. If S is removed, then $\mathcal{L}(G) = \emptyset$.
5. Remove any variable A not reachable from S.
6. Remove any terminal which does not appear in some rule.

Back to finiteness of CFGs

Question 14

Given a CFG G, is $|\mathcal{L}(G)|$ finite?

Algorithm 15

1. Remove redundant variables and terminals.
2. Turn into a CNF.
3. Create a graph C:
   - Node for each variable $v \in V$.
   - **Directed** edges $(A, B), (A, C)$ for each rule $A \to BC \in R$.
4. Return `TRUE` iff C has a no cycles.

Correctness?

Less efficient algorithm "using" the pumping lemma:

- $|\mathcal{L}(G)| = \infty$ iff $\exists w \in \mathcal{L}(G)$ with $\ell \leq |w| \leq 2\ell$ (?) (ℓ is the pumping length)
- Hence, it suffices to check whether $\mathcal{L}(G)$ has a word on such length

Inherent ambiguity

Question 16

Given a CFG G, is $\mathcal{L}(G)$ inherently ambiguous?

(i.e., for any CFG generating $\mathcal{L}(G)$ exists $w \in \mathcal{L}$ with two different parse trees).

Fact 17

There is no algorithm to solve CFG inherent ambiguity.

We will not prove this fact, yet you want to know it to put things in context.

When are two CFGs equivalent?

Question 18

Given two CFG G_1 and G_2, test if $L(G_1) = L(G_2)$.
Is there an algorithm to solve this problem?

A short summary

- Regular Languages $\equiv$ Finite Automata.
- Context Free Languages $\equiv$ Push Down Automata.
- Closure properties of regular languages and of CFLs.
- Most algorithmic problems for finite automata are solvable.
- Some algorithmic problems for finite automata are not solvable.
- Pumping lemmata for both classes of languages.
- There are additional languages out there.

View over the horizon