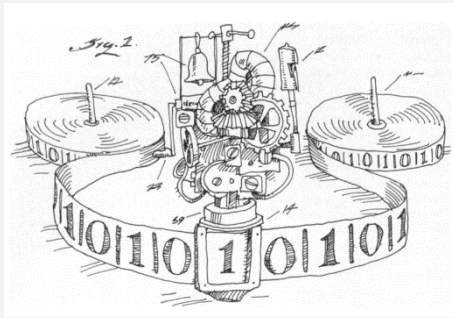
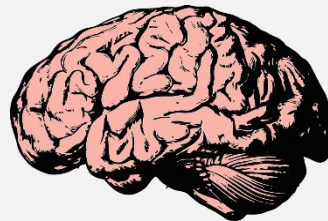
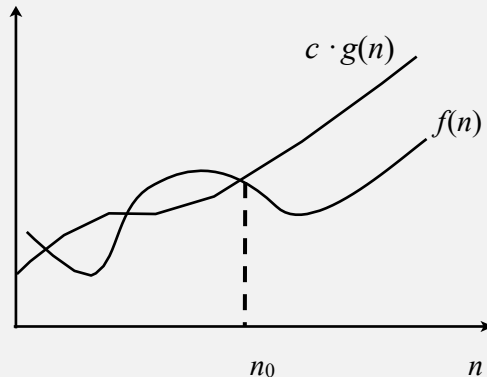


COMP 3602

Intro to Theory of Computation

Lecture 7

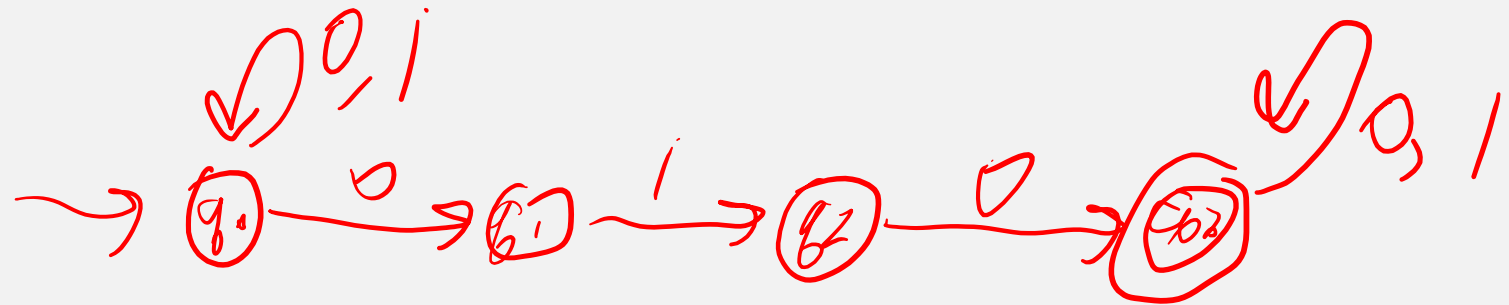


Logistics

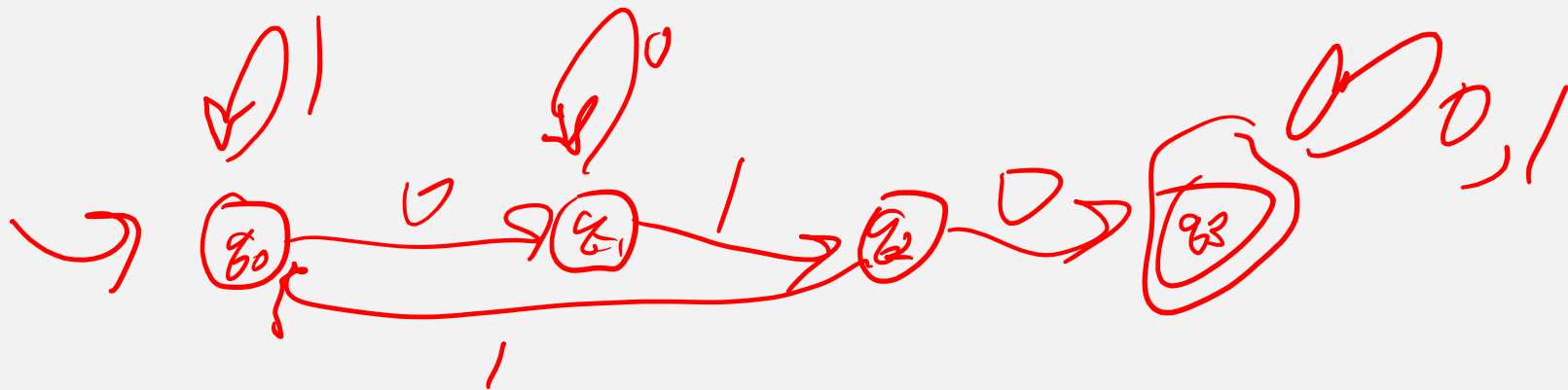
- Test 1 day/time: Mon Feb 2
 - Written test on Turing machines, automata/regex, PDA/CFGs, pumping lemmas
 - Practice problems are up
 - One **letter-size double-sided handwritten** cheatsheet allowed
- Theory reading group
 - Stays Mondays 1pm for now, sorry!
- Drills are up, some due Feb 2nd.
 - Do the sample drills first!
 - You can do the drills multiple times, but the grade is an average of all attempts
 - So if you open it and let it time out, it will add a 0 to your average!

Practice problems questions?

16. $\{w \in \{0,1\}^* \mid w \text{ contains subseq } 010\}$



NFA



DFA

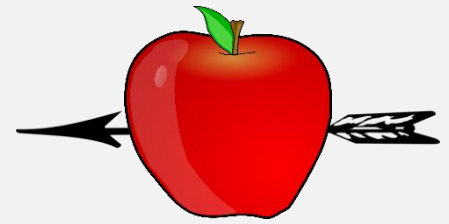
1. Give regular expressions and draw automata (DFAs or NFAs) for the following languages:
 - a. $\{w \in \{0,1\}^* \mid w \text{ consists only of the string } 010\}$
 - b. $\{w \in \{0,1\}^* \mid w \text{ contains a substring } 010\}$
 - c. $\{w \in \{0,1\}^* \mid w \text{ contains a substring } 010, \text{ starts with } 0 \text{ and ends with } 1\}$
 - d. $\{w \in \{0,1\}^* \mid w \text{ does not contain a substring } 010\}$
 - e. $\{w \in \{a,b,c\}^* \mid w \text{ contains exactly 2 "c" and at least one "a" }\}$
 - f. $\{w \in \{a,b,c\}^* \mid \text{length of } w \text{ is divisible by 5 and it ends at "a" }\}$
2. Give NFAs with the specified number of states for the following languages, then use the subset construction to get DFAs out of them.
 - a. $L = 1^*$, one state.
 - b. Strings with all 1s (if any) in the same substring (That is, strings without a substring 100^*1). Three states.
 - c. $0^*(1000^*)^*$ with three states (here, "1000*" means 100 followed by 0^*)
3. Give a pushdown automaton and a context-free grammar for the following languages (for extra practice, try writing grammars for languages above)
 - a. $\{w \in \{0,1\}^* \mid \text{the length of } w \text{ is odd}\}$
 - b. $\{w \in \{a,b,c\}^* \mid w \text{ contains more c's than a's}\}$
 - c. $\{w \in \{0,1\}^* \mid w \text{ is a palindrome or } w=0^n u 0^n \text{ for some } n, \text{ where } u \in \{0,1\}^* \}$
4. Show using the pumping lemmas that
 - a. $\{a^n b \{a,b,c,d\}^* c d^{2n}\}$ is not regular.
 - b. $\{w \in \{0,1\}^* \mid \text{contains a substring } 10 \text{ or } w=0^n 1^m \text{ where } n \neq m \}$ is not regular. Hint: use closure properties of regular languages.
 - c. $\{ww^r w \mid w = w_1 \dots w_n \in \{0,1\}^* \text{ where } w^r = w_n \dots w_1 \text{ is the reverse of } w \}$ is neither regular nor context-free.
 - d. $\{1^{n^2}\}$ is neither regular nor context-free.

5. Describe Turing machines for the following languages. Use the definition of Turing machines with a single 2-way infinite tape, and movement directions $\{L,R\}$. Make sure to include both the informal outline of the algorithm, and the full description, that is, Σ , Γ , Q , q_0 , q_a , q_r and δ . Do not draw diagrams for Turing machines! Hint: first look at a few examples of strings that are in the language and strings that are not in the language, and don't forget the empty string.
 - $L_1 = \{w \in \{b,c,d\}^* \mid \text{non-empty strings of length at least 3 with the last three symbols all different.}\}$
 - $L_2 = \{w \in \{0,1\}^* \mid w=0^n1^n0^{2n} \text{ for } n \geq 0 \}$. That is, L_2 consists of all strings of length divisible by 4 (including length 0), where the first quarter of the string is all 0s, second quarter is all 1s, and the second half all 0s again.
6. Describe a Turing machine that decrements by 2 a natural number ≥ 2 written on its tape in binary. It should end in q_a pointing to the first symbol of the output if the tape contained a string starting with 1 encoding a natural number $x \geq 2$, and in q_r otherwise. Give both an informal algorithm and a full description of the TM.
7. Consider a TM where instead of "move left one cell" and "move right one cell" the only available moving operations are "jump left 11 steps", "jump right 5 steps" and "stay at the same place"; let's call this model 11L5RS Turing machines. Just as in the model of TM we defined in class, 11L5RS TMs have one two-way-infinite tape, the only difference is the movement operations.
 - Show that 11L5RS TM model is equivalent to the model we defined in class. Do both directions, with constructions (from M construct M'), and an outline of a proof why your construction works.
 - Show that if instead it were "jump left 10 steps" "jump right 5 steps", then this 10L5RS TM model would not be able to simulate the standard TM.
8. Show that single-tape Turing machines that only move right recognize exactly the regular languages. Hint: do both directions of the equivalence, and remember that the Turing machine is not guaranteed to stop.
9. Show that finite automata with two stacks can simulate Turing machines.

Do not draw diagrams for Turing machines!

Questions/comments?

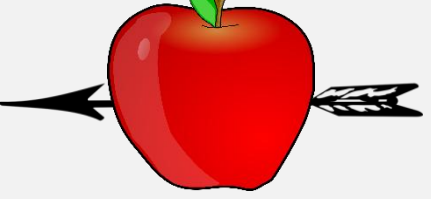
Context-free grammars



A special kind of grammars is **context-free grammars**, where symbols have the same meaning wherever they are.

A context-free grammar consists of

- A set V of **variables** (using capital letters), including a **start variable** S .
 - Note: these are variables, not sets.
- A set Σ of **terminals** (disjoint from V ; alphabet of the language.)
- A set R of **rules**, where each rule consists of a variable from V , followed by $\rightarrow$, followed by a string of variables and terminals.
 - This is not the same “ $\rightarrow$ ” as implication, a different use of the same symbol.
 - If several rules start with the same non-terminal such as $A \rightarrow w_1$, $A \rightarrow w_2$, etc, we use a shortcut “ $|$ ” (means “or”) to write them as one rule: $A \rightarrow w_1 | w_2 | \dots | w_k$



Context-free grammars

A context-free grammar: a set V of **variables**, including a **start variable** S , a set Σ (Sigma) of **terminals** (alphabet) and a set R of **rules** of the form $A \rightarrow w$, where $A \in V, w \in (V \cup \Sigma)^*$

- If $A \rightarrow w$ is a rule, we say variable A **yields** string w .
 - Can use A within the rule, as many times as we want: recursion!
 - Different occurrences of the same variable can produce different strings.
- A **derivation** is a sequence of strings each of which is obtained from the previous by applying some rule to its substring.
 - If $w = \ell Ar$ and there is a rule $A \rightarrow u$, then ℓur is **directly derived** from w , written $w \Rightarrow \ell ur$.
 - w_n is **derived from** w_0 if there is a sequence $w_0, w_1, \dots, w_n$ where $\forall i \ w_i \Rightarrow w_{i+1}$
- **A language generated by a grammar** consists of all strings of terminals that can be derived from the start variable S .

Pumping lemma for context-free languages

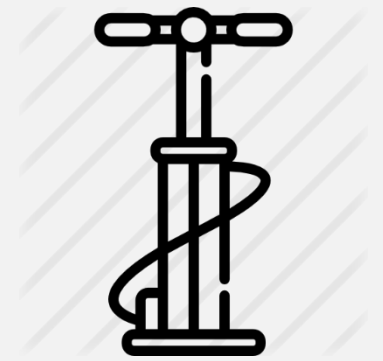
- *Pumping lemma:* If L is a **context-free** language, then there exists a number p (pumping length) such that every string s with $|s| \geq p$ can be written as a concatenation of strings u, v, x, y, z ($s = uvxyz$) where
 1. For each $i \geq 0$, string $uv^i xy^i z \in L$
 2. $|vy| > 0$
 3. $|vxy| \leq p$

Writing the statement with quantifiers:

$\exists p \in \mathbb{N} \ \forall s \in L (|s| \geq p \rightarrow$

$\exists u, v, x, y, z \ s = uvxyz \wedge \forall i \ uv^i xy^i z \in L \wedge |vy| > 0 \wedge |vxy| \leq p)$

Proof idea: a non-terminal must repeat on a path when the parse tree is large enough .





- To prove that L is not context-free
- We pick a string $s \in L$ that depends on a parameter p
- Show that for any split of s in u, v, x, y, z such that $|vy| > 0$ and $|vxy| \leq p$, there is an $i \geq 0$ with $uv^i xy^i z \notin L$
- Example: $\{a^n b^n c^n \mid w \in \{a, b, c\}^*\}$
 - Let $s = a^p b^p c^p$
 - Similar to $0^p 1^p$ for proving $\{0^n 1^n\}$ is non-regular
 - Let $s = uvxyz$, with $|vy| > 0$
 - Case 1: each of v, y contains at most one type of symbol (all a's, all b's, all c's)
 - Then pumping does not change the third type
 - Case 2: at least one of v, y contains a mix
 - Then $uv^2 xy^2 z$ has symbols in wrong order.



- So $\{a^n b^n c^n \mid w \in \{a, b, c\}^*\}$ is not context-free
 - But $\{a^i b^j c^k \mid i = j\}$ and $\{a^i b^j c^k \mid i = k\}$ are context-free!
- *Corollary:* Context-free languages are not closed under intersection and complementation.
 - Proof: for intersection, the counterexample above.
 - For complementation:
 - They are closed under union (do ε -arrows at the new start, as for regular)
 - If they were closed under complementation, could have done DeMorgan's $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ to get intersection. But we know we cannot get intersection by above.



- To prove that L is not context-free
 - We pick a string $s \in L$ that depends on a parameter p
 - Show that for any split of s in u, v, x, y, z such that $|vy| > 0$ and $|vxy| \leq p$, there is an $i \geq 0$ with $uv^i xy^i z \notin L$
-
- Example: $\{ww \mid w \in \{0,1\}^*\}$
 - Let $s = 0^p 1^p 0^p 1^p$
 - Before we used $0^p 10^p 1$. But it does not work! With $v = 0, x = 1, y = 0$ can pump.
 - Let $s = uvxyz$, with $|vy| > 0$
 - vxy should have the middle of the string
 - If vxy are all in the first half, then pumping it would move 1s in front of 2nd w.
 - Or make the first w only 0s, while second half has 1s.
 - Same for in the second half.
 - Suppose we pump down s ; since $|vxy| \leq p$, get $0^p 1^i 0^j 1^p$ with $i \neq p$ or $j \neq p$
 - Thus this string is not of the form ww .

Pushdown automata: NFA+stack



- Pushdown automaton (PDA):
 - Input alphabet Σ
 - **Stack alphabet Γ**
 - Set of states Q
 - Start state $q_0 \in Q$
 - Set of final (accepting) states $F \subseteq Q$
 - Transition function $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$
- Computation of a PDA M on a string $w = w_1 \dots w_n$:
 - On each step M reads a symbol from the input, pops a symbol off the stack and then changes state and pushes a symbol on the stack
 - In all three of read, pop, push, the “symbol” may be empty, that is, ε .
 - Popping an empty stack, in particular, always returns ε
 - Often want to put an end marker $\$$ into the stack in the first transition to detect empty.
 - M accepts w if there exist sequences of states $r_0 \dots r_m$ **and stack content strings** $s_0 \dots s_m$ with $m \geq n$, $r_0 = q_0$, $s_0 = \lambda$, $r_m \in F$ and all intermediate states and stacks computed correctly.

Context-free languages = recognized by PDAs

- From a grammar to a PDA:
 - Match terminals right away
 - Keep the remaining string (starting with first non-terminal) on the stack
 - Guess what rule to use with non-determinism

From PDA to grammar

- Idea: make a variable for every pair of states: $V = \{A_{pq} \mid p, q, \in Q\}$
 - It will generate strings processed by the PDA between these states

From a PDA to a grammar

- Let M be a PDA.  Without loss of generality
 - Assume WLOG that each transition either pushes or pops, but not both.
 - And that $|F| = \{q_a\}$, and at accept the stack is empty.
- Make a variable for every pair of states: $V = \{A_{pq} \mid p, q \in Q\}$
- Start variable A_{q_0, q_a}
- Rules:
 - 1. Split into separate strings: for every triple of states add a rule $A_{pq} \rightarrow A_{pr}A_{rq}$
 - When the stack is same at all three of p, q, r
 - 2. Push/pop pair: If $(r, u) \in \delta(p, a, \varepsilon)$ and $(q, \varepsilon) \in \delta(s, b, u)$, add $A_{pq} \rightarrow aA_{rs}b$
 - 3. Add $A_{pp} \rightarrow \lambda$ for all $p \in Q$