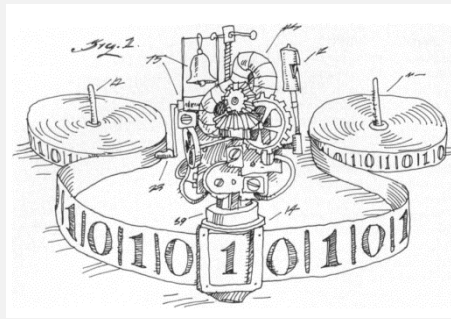
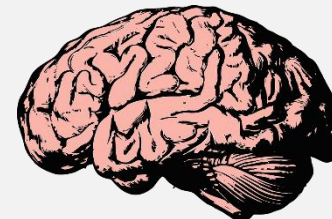
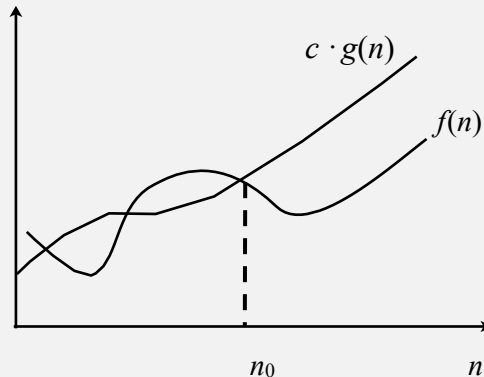


COMP 3602

Intro to Theory of Computation

Lecture 5



Logistics

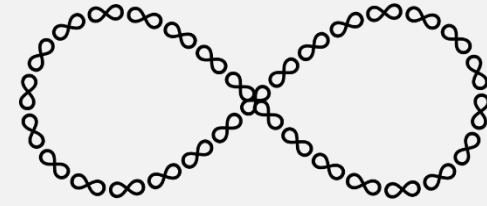
- Test 1 day/time?
 - Written test on Turing machines, automata and today's (maybe Monday's) material

Mon Feb 2?

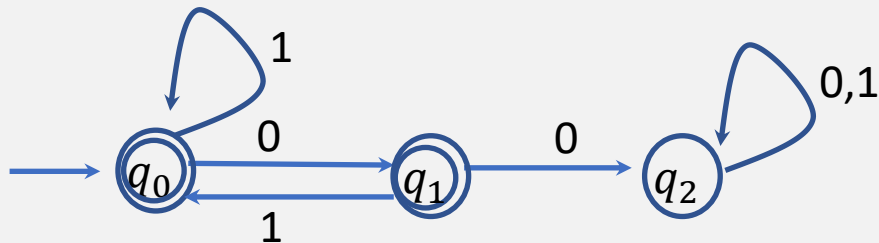
- Theory reading group
 - From next week Monday 1pm, in the seminar room EN-2022
- Drills are up
 - You can do Turing machines and regular expressions drills by now
 - Do the sample drills first!
 - You can do the drills multiple times, but the grade is an average of all attempts
 - So if you open it and let it time out, it will add a 0 to your average!

Questions/comments?

Structure of regular languages



- If a language is finite, it must be regular
 - Just make DFAs for every string in the language and take their union.
- What kind of structure do infinite regular languages have?
 - Hint: the only regular operation that can create an infinite language out of a finite one is the Kleene star.



Pumping lemma for regular languages

- *Pumping lemma*: If L is a regular 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 x, y, z ($s = xyz$) where
 1. For each $i \geq 0$, string $xy^iz \in L$
 - y^i is $yyyyy \dots y$ written i times. $y^0 = \lambda$
 2. $|y| > 0$
 3. $|xy| \leq p$

Writing the statement with quantifiers:

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

$\exists x, y, z \ s = xyz \wedge \forall i \ xy^iz \in L \wedge |y| > 0 \wedge |xy| \leq p)$



Pumping Lemma: If L is a regular 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 x, y, z ($s = xyz$) where

1. For each $i \geq 0$, string $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

Proof: If L is regular, some DFA D recognizes it.

- Let $p = |Q|$ be the number of states in D .
 - Let $s \in L$ be a string with $|s| \geq p$.

Pumping lemma: If L is a regular 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 x, y, z ($s = xyz$) where

1. For each $i \geq 0$, string $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

Proof: If L is regular, some DFA D recognizes it.

- Let $p = |Q|$ be the number of states in D .
 - Let $s \in L$ be a string with $|s| \geq p$. The computation of D on s consists of p transitions, and so $p + 1$ state names (including the initial state).
 - As there are only p states in D , at least one state occurs twice in this computation. Let's say state q is the state that repeats first, and it occurs at steps i and then j .
 - That means there is a loop in D starting and ending at q .
 - All states from the start to one at step $j - 1$ are different.
 - Take x to be the part of s processed before getting to step i , y to be part of s processed between steps i and j , and z from j to the end of s .

Pumping lemma: If L is a regular 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 x, y, z ($s = xyz$) where

1. For each $i \geq 0$, string $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

Proof: If L is regular, some DFA D recognizes it.

- Let $p = |Q|$ be the number of states in D .
 - Let $s \in L$ be a string with $|s| \geq p$. The computation of D on s consists of p transitions, and so $p + 1$ state names (including the initial state).
 - As there are only p states in D , at least one state occurs twice in this computation. Let's say state q is the state that repeats first, and it occurs at steps i and then j .
 - That means there is a loop in D starting and ending at q .
 - All states from the start to one at step $j - 1$ are different.
 - Take x to be the part of s processed before getting to step i , y to be part of s processed between steps i and j , and z from j to the end of s .
- 1: Now, $xy^iz \in L$ since the same sequence of states without steps i to $j - 1$ is accepting, and so is a sequence with steps i to $j - 1$ repeated several times
 - As this is a DFA, D will always go through the same sequence of states starting from q on a y .
- 2: Since D is a DFA, there is at least one symbol of s read between steps i and j , so $|y| > 0$
- 3: Since all states until the first repetition of q are different, $|xy| \leq p$

Pumping lemma: If L is a regular 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 x, y, z ($s = xyz$) where

1. For each $i \geq 0$, string $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

Proof: If L is regular, some DFA D recognizes it.

- Let $p = |Q|$ be the number of states in D .
 - Let $s \in L$ be a string with $|s| \geq p$. The computation of D on s consists of p transitions, and so $p + 1$ state names (including the initial state).
 - Take x to be the part of s processed before getting to step i , y to be part of s processed between steps i and j , and z from j to the end of s .
- 1: Now, $xy^iz \in L$ since the same sequence of states without steps i to $j - 1$ is accepting, and so is a sequence with steps i to $j - 1$ repeated several times
 - As this is a DFA, D will always go through the same sequence of states starting from q on a y .
- 2: Since D is a DFA, there is at least one symbol of s read between steps i and j , so $|y| > 0$
- 3: Since all states until the first repetition of q are different, $|xy| \leq p$



Proving that a language is not regular.

- **Pumping lemma:**

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

$\exists x, y, z \ s = xyz \wedge \forall i \ xy^i z \in L \wedge |y| > 0 \wedge |xy| \leq p)$

- Let's take a contrapositive:

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

$\exists x, y, z \ s = xyz \wedge \forall i \ xy^i z \in L \wedge |y| > 0 \wedge |xy| \leq p))$

then L is not regular.



- Pumping lemma: if L is regular, then

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

$$\exists x, y, z \ s = xyz \wedge \forall i \ xy^i z \in L \wedge |y| > 0 \wedge |xy| \leq p)$$

- Let's take a contrapositive:

$$\neg (\exists p \in \mathbb{N} \ \forall s \in L (|s| \geq p \rightarrow \exists x, y, z \ s = xyz \wedge \forall i \ xy^i z \in L \wedge |y| > 0 \wedge |xy| \leq p)) \\ \rightarrow L \text{ is not regular}$$

$\equiv$

$$(\forall p \in \mathbb{N} \ \exists s \in L (|s| \geq p \wedge \forall x, y, z (s \neq xyz \vee \exists i \ xy^i z \notin L \vee |y| = 0 \vee |xy| > p))) \\ \rightarrow L \text{ is not regular}$$

$\equiv$

$$(\forall p \in \mathbb{N} \ \exists s \in L (|s| \geq p \wedge \forall x, y, z ((s = xyz \wedge |y| > 0 \wedge |xy| \leq p) \rightarrow \exists i \ xy^i z \notin L))) \\ \rightarrow L \text{ is not regular}$$

Suppose that for any p there is a string $s \in L$ with $|s| \geq p$ so that no matter how s is split into x, y, z with $|y| > 0$ and $|xy| \leq p$, there is an $i \geq 0$ such that the “pumped” string $xy^i z$ is not in L . Then L is not regular.



Proving that a language is not regular.

- Pumping lemma: Suppose that for any p there is a string $s \in L$ with $|s| \geq p$ so that no matter how s is split into x, y, z with $|y| > 0$ and $|xy| \leq p$, there is an $i \geq 0$ such that the “pumped” string xy^iz is not in L . Then L is not regular.
 - To prove that L is not regular
 - We pick a string $s \in L$ that depends on a parameter p
 - Show that for any split of s in x, y, z with $|y| > 0$ and $|xy| \leq p$, there is an $i \geq 0$ with $xy^iz \notin L$



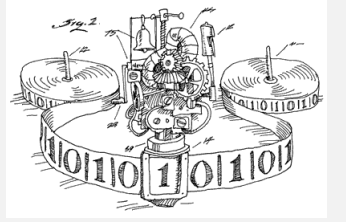
- Pumping lemma: Suppose that for any p there is a string $s \in L$ with $|s| \geq p$ so that no matter how s is split into x, y, z with $|y| > 0$ and $|xy| \leq p$, there is an $i \geq 0$ such that the “pumped” string xy^iz is not in L . Then L is not regular.
 - To prove that L is not regular
 - We pick a string $s \in L$ that depends on a parameter p
 - Show that for any split of s in x, y, z with $|y| > 0$ and $|xy| \leq p$, there is an $i \geq 0$ with $xy^iz \notin L$
- Example: $\{0^n 1^n \mid n \geq 0\}$
 - Consider a string $0^p 1^p$.
 - Let $s = xyz$, with $|y| > 0$
 - Then y consists of either only 0s, or only 1s, or 0s followed by 1s.
 - In the first two cases, repeating or removing y changes only the number of one of symbols, but not the other. So get something not in L
 - In the last case, repeating y results in 0s after 1s. Also not in L
 - Using $|xy| \leq p$ condition, get that y must consist of just 0s, so only need the first case.



- To prove that L is not regular
 - We pick a string $s \in L$ that depends on a parameter p
 - Show that for any split of s in x, y, z such that $|y| > 0$ and $|xy| \leq p$, there is an $i \geq 0$ with $xy^iz \notin L$
- Example: $\{ww \mid w \in \{0,1\}^*\}$
 - Consider a string $0^p 1 0^p 1$.
 - Let $s = xyz$, with $|y| > 0$
 - Using $|xy| \leq p$ condition, get that y must consist of just 0s
 - Then xy^iz would have extra 0s in the first part (or fewer if $i = 0$)
 - Note: not every string works! A string $(01)^{2p}$ can be pumped
 - take $y = 0101, x = \lambda, z = (01)^{2p-2}$.



These non-regular languages seem simple!



- Can we characterize languages like $\{0^n 1^n\}$ in a similar way to regular languages?



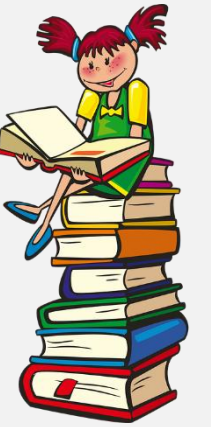
Pushdown automata: NFA+stack



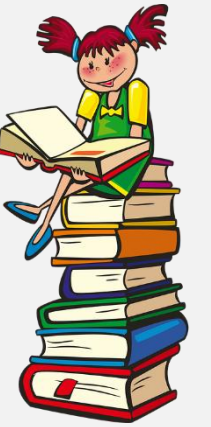
- Pushdown automaton (PDA):
 - Input alphabet Σ
 - **Stack alphabet Γ**
 - Set of states Q
 - Start state $q_0 \in Q$
 - Set of final (accepting) states $F \subseteq Q$
 - Transition function $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$
- 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
 - 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.

Pushdown automata: NFA+stack

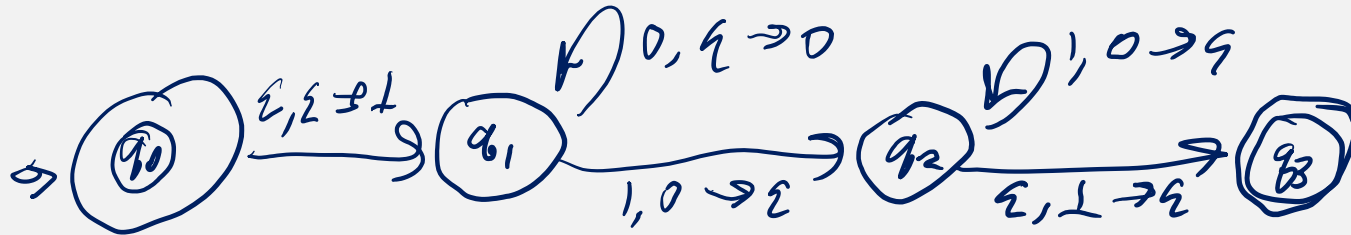
- Pushdown automaton (PDA): $(\Sigma, \Gamma, Q, q_0, F, \delta)$
 - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$
- Example: recognizing $\{0^n 1^n\}$
 - Idea: put all 0s on stack, then match them with 1s.



Pushdown automata: NFA+stack

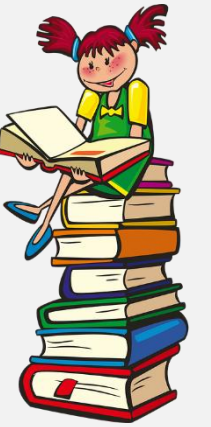


- Pushdown automaton (PDA): $(\Sigma, \Gamma, Q, q_0, F, \delta)$
 - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$
- Example: recognizing $\{0^n 1^n\}$
 - Idea: put all 0s on stack, then match them with 1s.

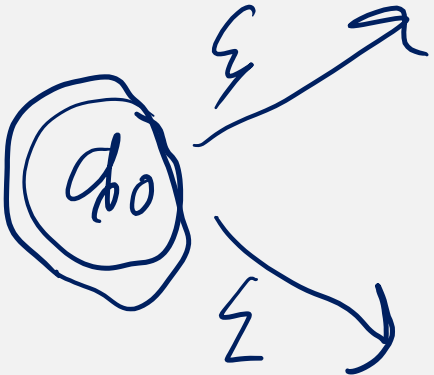


$N_{(0,1)}$

Pushdown automata: NFA+stack



- Pushdown automaton (PDA): $(\Sigma, \Gamma, Q, q_0, F, \delta)$
 - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$
- Example: recognizing $\{a^i b^j c^k \mid i = j \text{ or } i = k\}$
 - Similar, with ε -arrows to choose between matching b's or c's to a's.



- Need non-determinism here!
 - Can prove that deterministic PDAs cannot recognize this language.

Puzzle

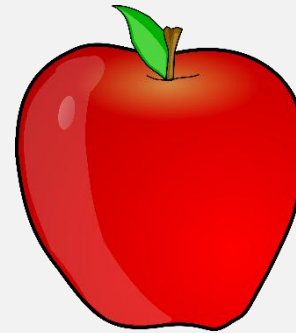
- Are the following English sentences built the same way?



– Time flies like an arrow.

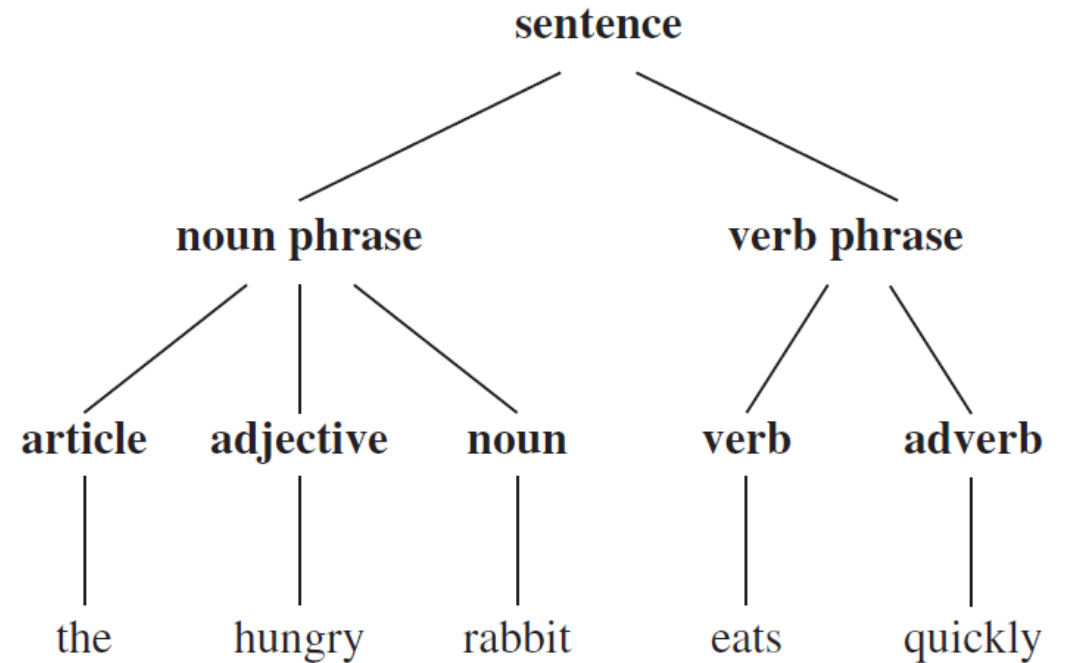


– Fruit flies like an apple.



Grammars

- Remember that sets of strings are called **languages**.
- A type of recursive definition of a language is called a **grammar**.
- Different natural languages also have different grammars!
 - English: Subject/Verb/Object
 - Japanese: Subject/Object/Verb
 - Gaelic: Verb/Subject/Object
 - Ukrainian: order does not matter



Formal grammars: Noam Chomsky

- Generative grammar: describes how to generate strings in a language
 - Similar idea to regular expressions.
 - Also called rewriting system
- Chomsky's grammars:
 - Alphabet Σ (called "*terminals*")
 - A set finite set V of *variables* (also called "non-terminals")
 - usually denoted by capital letters
 - A special *start variable* $S \in V$
 - A set of *production rules* R
 - Each rule is of the form $string_{left} \rightarrow string_{right}$, where $string_{left}, string_{right} \in (\Sigma \cup V)^*$
 - With an additional requirement that $string_{left}$ contains at least one variable symbol

