Indiv HW4 due tomorrow
Group HW4 available

CSE 105
THEORY OF COMPUTATION

"Winter" 2018

http://cseweb.ucsd.edu/classes/wi18/cse105-ab/

Today's learning goals

Sipser Section 2.3, 3.1

- Give examples of non context-free-languages
- State and use closure properties of regular and context-free languages
- Determine if a given language is regular, nonregular but context-free, or not-context-free
- Identify the components of a formal definition of a Turing machine
- Trace the computation of a Turing machine on given input

Classifications of languages

A language L is

Regular iff it is described by some regular expression
iff it is recognized by some DFA
iff it is recognized by some NFA

Context-free iff it is recognized by some (nondet) PDA
iff it is generated by some (ambiguous) CFG

What's left?

- **Fact**: Every regular language is a context-free language.
- **Fact**: There are context-free languages that are nonregular. eg. $\{0^n 1^n \mid n > 0\}$
- **Fact**: There are countably infinitely many regular languages.
- **Fact**: There are countably infinitely many context-free languages.

But uncountably many languages

Most languages are not context-free languages!

Informal intuition

There must be at least one language that is not context-free

Which specific language is not context-free?

A. $\{ 0^n 1^m 0^n \mid m,n \geq 0 \}$

B. $\{ 0^n 1^n 0^n \mid n \geq 0 \}$

C. $\{ 0^n 1^{2n} \mid n \geq 0 \}$

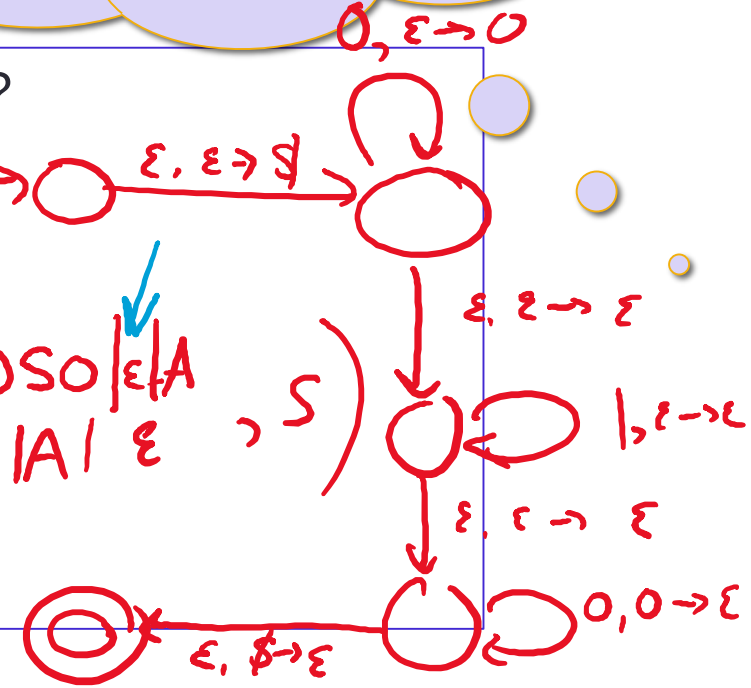
D. $\{ 0^n 1^{2m} \mid m,n \geq 0 \}$ reg

E. I don't know.

CFG $S \to 0S1 \mid \varepsilon$

PDA

CFG $(\{S,A\}, \{0,1\}, S, S \to 0S0 \mid \varepsilon \mid A,\ A \to 1A \mid \varepsilon)$



Examples of non-context-free languages

- $\{ a^n b^n c^n \mid 0 \leq n \}$ Sipser Ex 2.36
- $\{ a^i b^j c^k \mid 0 \leq i \leq j \leq k \}$ Sipser Ex 2.37
- $\{ w\,w \mid w \text{ is in } \{0,1\}^* \}$ Sipser Ex 2.38

compare to $\{ww^R \mid w \text{ is in } \{0,1\}^*\}$ CFL

To prove… Pumping lemma for CFLs (won't cover in CSE 105)

Closure properties of …

The class of regular languages is closed under

- Union
- Concatenation
- Star
- Complementation
- Intersection
- Difference
- Reversal
- FlipBits

The class of context-free languages is closed under

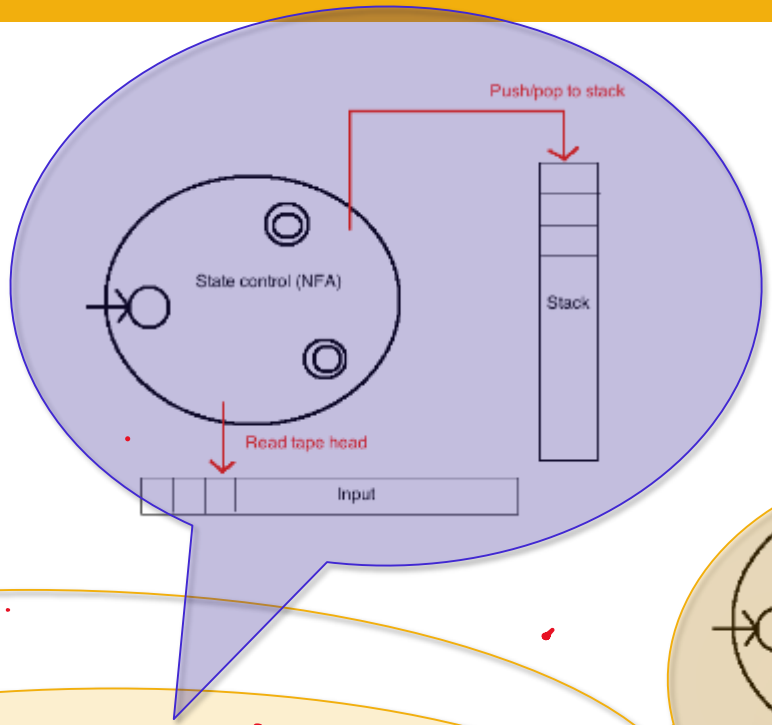
- Union
- Concatenation
- Star
- Reversal
- FlipBits

The class of context-free languages is not closed under

- Intersection
- Complementation
- Difference

Languages over Σ

???



???

Context-free languages

$\{a^n b^n \mid n \geq 0\}$

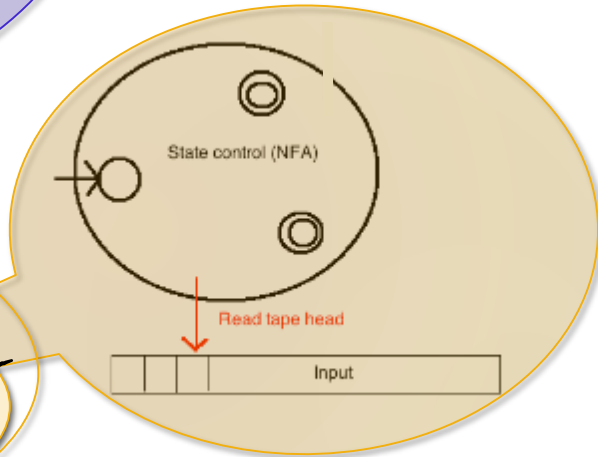
$\{a^n b^n \mid n > 3\}$

$L(a^* b^*)$

Regular languages

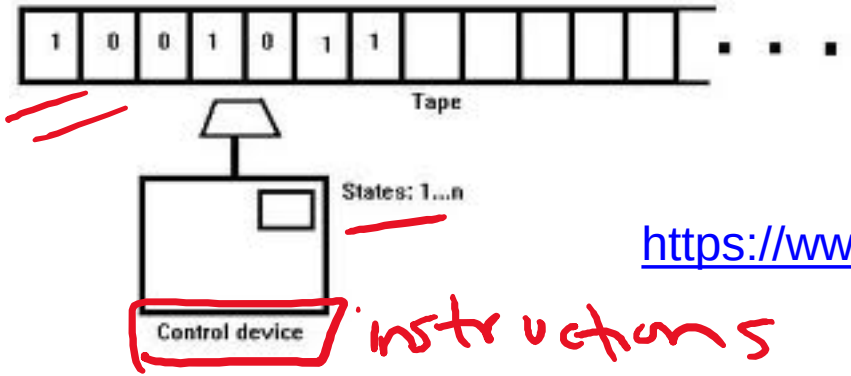
$\emptyset$

Σ^*



Turing machines

- Unlimited input
- Unlimited (read/write) memory
- Unlimited time



https://www.youtube.com/watch?v=eWq5wAX8K8A

Why is this model relevant?

- Division between program (CPU, state space) and data (memory) is a cornerstone of all modern computing
- Unbounded memory is outer limits of what modern computers (PCs, quantum computers, DNA computers) can implement.
- Simple enough to reason about (and diagonalize against), expressive enough to capture modern computation.

Formal definition of TM

CAUTION: Use Sipser Convention

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where Q, Σ, Γ are all finite sets and

1. Q is the set of states (finite)
2. Σ is the input alphabet (not containing blank symbol)
3. Γ is the tape alphabet (including blank symbol as well as all symbols in Σ)
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function
5. $q_0 \in Q$ is the start state
6. $q_{\text{accept}} \in Q$ is the accept state
7. $q_{\text{reject}} \in Q$ is the reject state

$\Sigma \subseteq \Gamma$

L, R

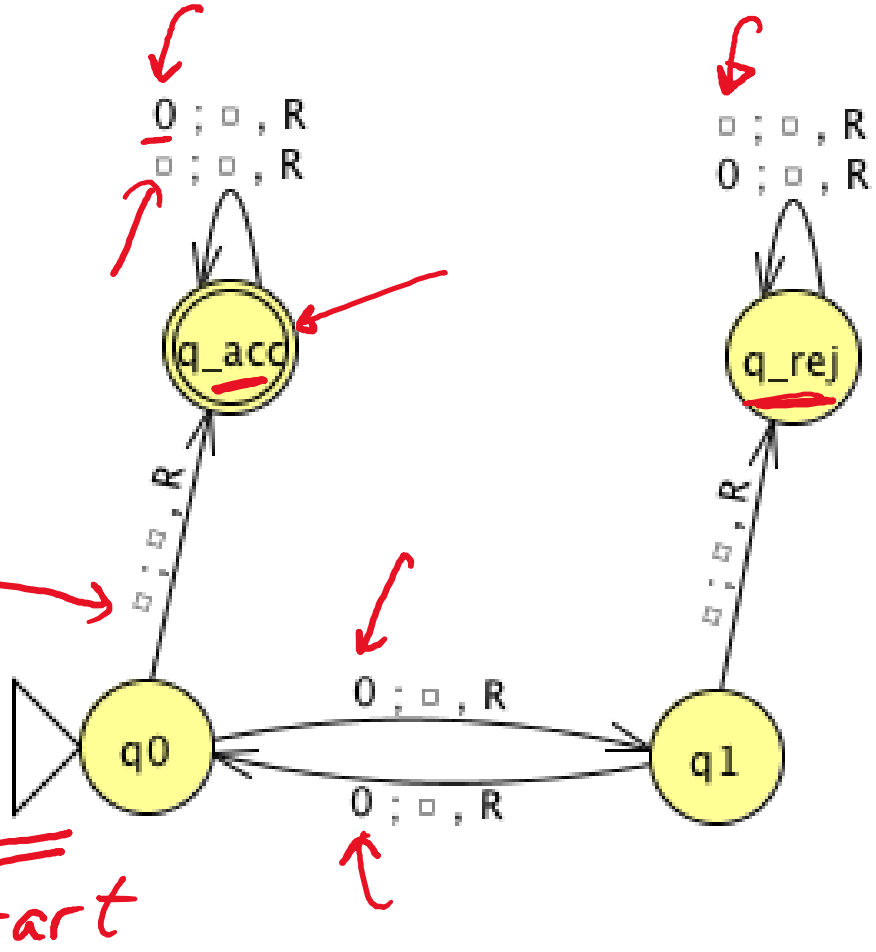
(state, symbol read from tape) → (state, symbol to write to tape, move on tape)

$q_{\text{reject}} \neq q_{\text{accept}}$

JFLAP

Σ = {0}
Γ = {0, ␣}

blanks



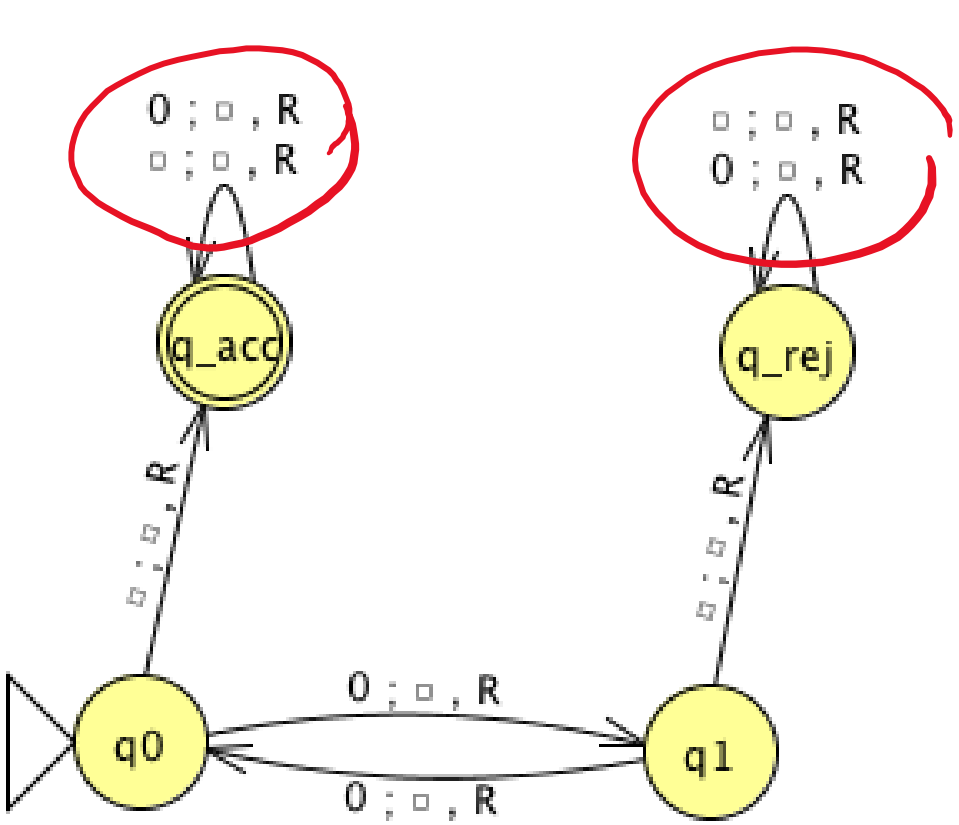
Turing machine computation

- Read/write head starts at leftmost position on tape
- Input string written on leftmost squares of tape, rest is blank
- Computation proceeds according to transition function:
  - Given current state of machine, and current symbol being read
  - the machine
    - transitions to new state
    - writes a symbol to its current position (overwriting existing symbol)
    - moves the tape head L or R
- Computation ends **if and when** machine enters either the **accept** or the **reject** state.

Language of a Turing machine

L(M) = { w | computation of M on w halts after entering the **accept** state}

i.e. L(M) = { w | w is accepted by M}



δ is well def

Language of a Turing machine

L(M) = { w | computation of M on w halts after entering the **accept** state}

i.e. L(M) = { w | w is accepted by M}

Comparing **TM**s and **PDA**s, which of the following is true:

A. Both TMs and PDAs may accept a string before reading all of it.
B. A TM may only read symbols, whereas a PDA may write to its stack.
C. Both TMs and PDAs must read the string from left to right.
D. States in a PDA must be either accepting or rejecting, but in a TM may be neither.
E. I don't know.

To think about

- Given a DFA, how would you simulate it with a TM?
- Given an NFA, how would you simulate it with a TM?
- Given a PDA, how would you simulate it with a TM?

For next time

IndivHW4 due Tuesday, February 13

For Monday, pre-class reading: pp. 166-167