CSE 105
THEORY OF COMPUTATION

"Winter" 2018

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

Today's learning goals

Sipser Section 2.2

- Design push-down automata to recognize specific languages
- Determine whether a language is recognizable by a (D or N) FA and/or a PDA

Reminders:

Group hw3 due Saturday

Exam 1 on Wednesday

Practice exam on class website

Push-down automaton

Sipser p. 111

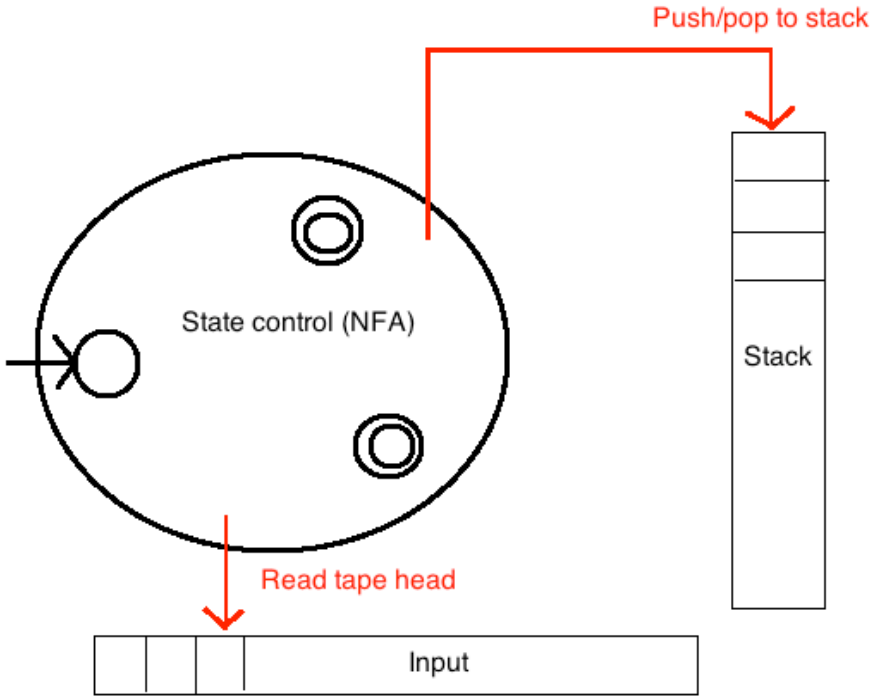
NFA + stack

- *Accept a string iff*

there is **some** sequence of states and **some** sequence of stack contents which processes the entire input string and ends in an accepting state.

- *Transitions* labelled a, b → c

mean "read an a from the input, pop b from the stack, push c to the stack"



Formal definition of PDA

Sipser Def 2.13 p. 113

A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, F are all finite sets and

1. Q is the set of states
2. Σ is the input alphabet
3. Γ is the stack alphabet
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accept states.

Informal description of PDA

Sipser p. 115

Step-by-step description of how computations process input, without drawing state diagram.

Should be able to reconstruct state diagram precisely from descripton.

Conventions:

- Can "test for end of stack" without providing details

  How? We can always push the end-of-stack symbol, $, at the start.
- Can "test for end of input" without providing details

  How? Can transform PDA to one where accepting states are only those reachable when there are no more input symbols.

Design sequence

Given language L, to build a PDA recognizing it

1. **Understand L**: e.g. Identify strings in L, strings not in L
2. **Sketch** high-level algorithm that determines membership in L
3. Translate to **informal description** of PDA
4. Translate to **state diagram** and/or formal definition of PDA
5. **Confirm** consistency with test cases.

Example

$\Sigma = \{ a, b, c \}$

$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$

Which of the following strings are **not** in L?

A. b $\in L$ $i=0$ $j=1$ $k=0$

B. abc $\in L$ $i=j=k=1$

C. abbcc $i=1$ $j=k=2$

D. aabcc

E. I don't know. $\notin \Sigma^*$ as a string.

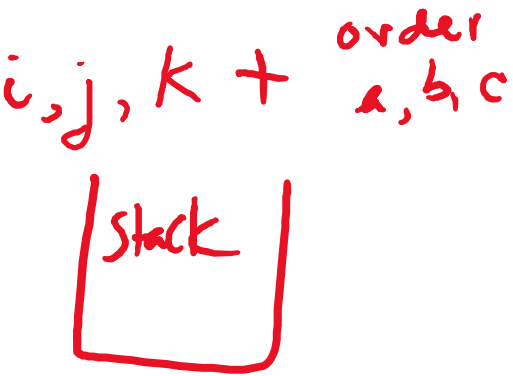
Designing a PDA

$$L = \{ a^i b^j c^k \mid i=j \text{ or } j=k, \text{ with } i,j,k \geq 0 \}$$

Informal description of PDA:

How would you design an algorithm that, given a string, decides if it is in this set?

- What information do you need to track?
- How much memory do you need?
- Are you using non-determinism?



Designing a PDA

$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$

Informal description of PDA:

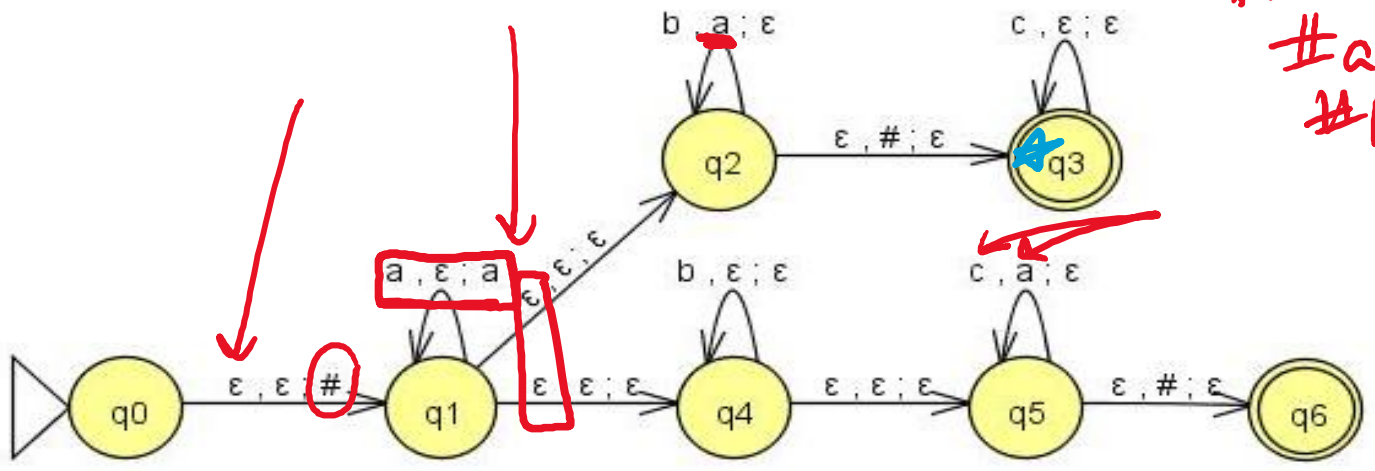
- The PDA pushes a # to indicate the top of the stack, then starts reading a's, pushing each one on to the stack.
- The PDA guesses when it's reached the end of the a's and whether to match the number of a's to the number of b's or the number of c's
- If trying to match number of b's with number of a's: PDA pops off a's for each b read. If there are more a's on the stack but no more b's being read, reject. When the end of the stack (#) is reached, the number of a's matches the number of b's. If this is the end of the input or if any number of c's is read at this point, accept; otherwise, reject.
- If trying to match the number of c's with number of a's: first read any number of b's without changing stack contents and then nondeterministically guess when to start reading c's. For each c read, pop one a off the stack. When the end of the stack (#) is reached the number of a's and c's so far match.

Designing a PDA

L is not a regular set

$L = \{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$

State diagram of PDA:



stuck

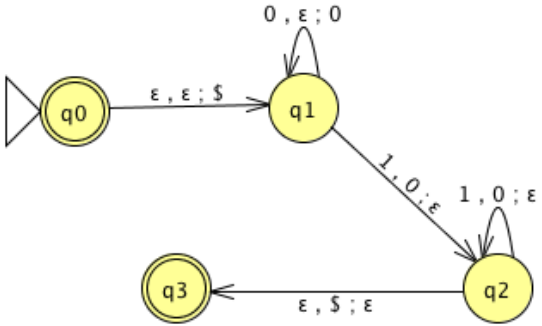
branch

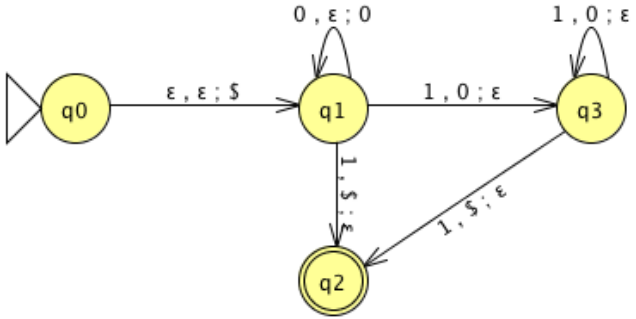
must have a at top of stack to follow this arrow

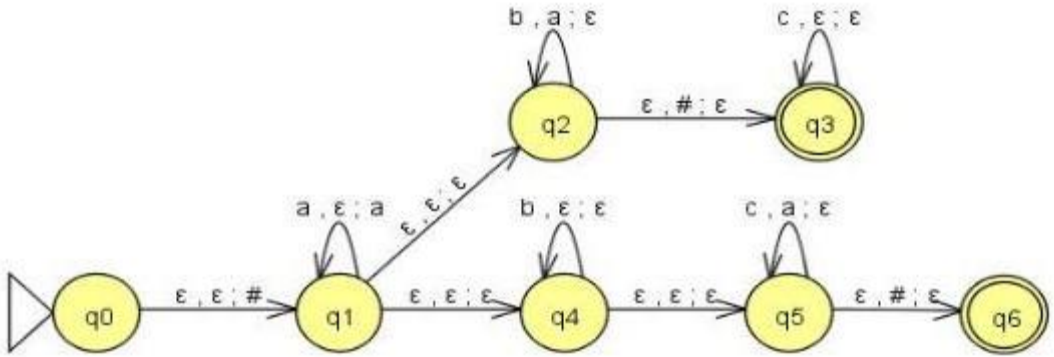
match #as to #bs

match #as to #cs

Examples so far







Another one?

$L = \{ a^i b^j c^k \mid i,j,k \geq 0 \}$

Actually, $L_1 \subseteq L$

Is there a PDA that recognizes L?

A. Yes, because it is a subset of a set we know is recognizable by a PDA, $\{ a^i b^j c^k \mid i=j \text{ or } i=k, \text{ with } i,j,k \geq 0 \}$
B. Yes, because L is regular
C. No, because L is regular
D. No, because L is not regular
E. I don't know.

$= L$!

so there's NFA N s.t. $L(N) = L$

use stack diagram N, ignore stack.

Regular sets and PDAs

Theorem: If L is regular then there is a PDA that recognizes it.

Pf: (Idea of previous slide)

Formally, ex...

Closure

The class of languages over Σ that are recognizable by PDA is closed under …

A. Complementation, by flipping accept/ reject states.

B. Union, by adding a spontaneous move from a new start state to the start states.

C. Concatenation, by adding spontaneous moves from accept states of one machine to the start state of the second.

D. Kleene star, by adding a fresh start state and spontaneous moves from accept states to the old start state.

E. I don't know.

uh oh

fixable

same issue

Closure

- Formal definition:

For next time

- Work on Group HW3 **due Saturday**

Exam 1 February 7 in class

- Practice exam on class website
- Review session on Monday
- Extra office hours on Google calendar on class website

Pre class-reading for Monday: pages 111-112.