$L = \{0^a 1^b \mid 0 < a < b\}$

- Keep track of the number of 0's and 1's
- Can't think of regex for L

Show that L is non-regular using...

Note A non-regular language could have a pumping length p.
Example:
$\{0^a 1^b 0^c \mid a,b,c \geq 0$ and if $a=1$ then $b=c\}$

Pumping Lemma

If L is a regular language, then there is a number p where, if s is any string in L of length at least p, then s may be divided into three pieces, s = xyz, such that:

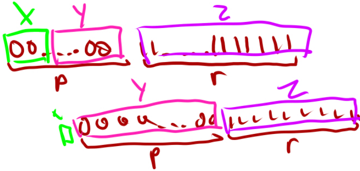
- $|y| > 0$
- $|xy| \leq p$
- for each $i \geq 0$, $xy^iz \in L$

If we can show that there's no possible value for p for L, then L is non regular.

xyz
→ xyyz
xyyyz

$r > p > 0$

When p is arbitrary, and $0 < p < r$, $s = 0^p 1^r \notin L$



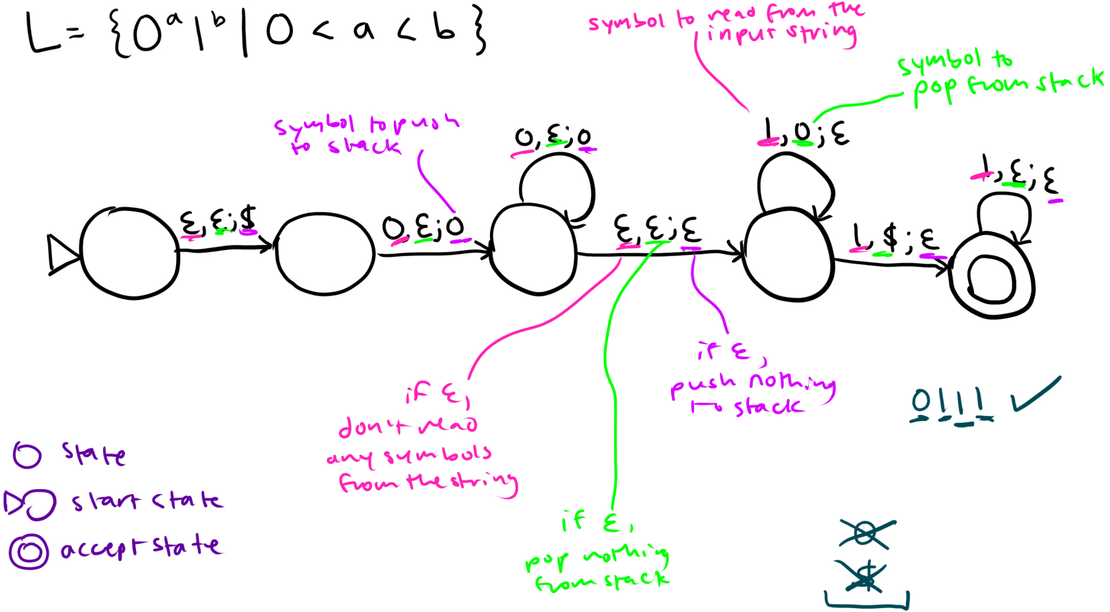
Since $|xy| \leq p$, x and y will both consist of all 0's.

For any possible $|y|$, xy^rz would result in a string that has at least as many 0's as 1's, which would not be in L.

$|S| > p$, but there is no possible way to divide s into xyz parts to satisfy all the statements above. So there is no p.

Push-Down Automata

$L = \{0^a 1^b \mid 0 < a < b\}$



CSE 105 Week 5 Discussion

Rachel Lim, Tony Hu

Regular or Non-regular?

- $A = \{ 0^k u 0^k \mid k > 0, u \in \Sigma^* \}$

- $B = \{ 0^k 1 u 0^k \mid k > 0, u \in \Sigma^* \}$

u absorbs unbalanced 0s

Regular or Non-regular?

001110

Σ^*

- $A = \{ 0^k u 0^k \mid k > 0, u \in \Sigma^* \}$

  Regular! A can be described by the regular expression $0\Sigma^*0$.

- $B = \{ 0^k 1 u 0^k \mid k > 0, u \in \Sigma^* \}$

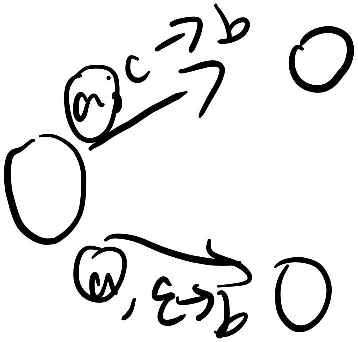
  Non-regular! Use the pumping lemma to prove a language is non-regular.

  Proof: Assume towards contradiction that B is regular and can be pumped. For any positive integer p, we can find a string $s = 0^p10^p \in B$ such that we can partition s into xyz, where $x = 0^a$, $y = 0^b$, $z = 0^c10^p$, $|xy| \leq p$, $|y| > 0$, and $a+b+c = p$. Let $i = 2$ so $xy^iz = 0^{p+b}10^p \notin B$ because $b>0$ so $p+b \neq p$. Hence, we have reached a contradiction and B is non-regular.

$p+b \neq p$

Pushdown Automata (PDA)

- Essentially a NFA with a stack (LIFO)
- Unlike DFAs/NFAs, PDAs has some non-constant memory to work with
- Each computation path when we use non-determinism will have its own stack
- PDAs can recognize both context free languages and regular languages



Formal Definition of PDA

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.

Transitions in PDAs

Assuming the alphabet = {a, b}, how do we interpret each of the following transitions:

- ε, a; b
- a, ε; b
- a, b; ε
- a, ε; ε
- ε, ε; a
- ε, a; ε
- ε, ε; ε
- a, b; a

Types of Transitions in PDAs

Assuming the alphabet = {a, b}, how do we interpret each of the following transitions:

- ε, a; b // Don't read any characters from the input, pop "a" from the stack, push "b" onto stack
- a, ε; b // Read "a" from the next character in input, don't pop from the stack, push "b" onto stack
- a, b; ε // Read "a" from the next character in input, pop "b" from the stack, don't push onto stack
- a, ε; ε // Read "a" from the next character in input, don't pop from and don't push onto stack
- ε, ε; a // Don't read any characters from the input, don't pop from the stack, push "a" onto stack
- ε, a; ε // Don't read any characters from the input, pop "a" from the stack, don't push onto stack
- ε, ε; ε // Don't read any characters from the input, don't pop from and don't push onto stack
- a, b; a // Read "a" from the next character in input, pop "b" from the stack, push "a" onto stack

Note: similar to NFAs, missing transitions result in "dead" computational paths

(cf. Sipser Exercises 2.6a/7) Consider the following informal description of a PDA:

> Read symbols from the input. As each a is read, push a 1 onto the stack. As soon as bs are seen, pop a 1 off the stack for each b read. If we reach the end of the string and the stack is nonempty, accept. If the stack becomes empty while we are reading bs, or if we find any as once we start reading bs, reject.

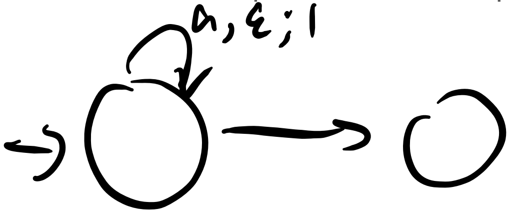
a) Draw the PDA that corresponds to the description above.

b) What is the language recognized by the PDA?

(*cf. Sipser Exercises 2.6a/7*) Consider the following informal description of a PDA:

> Read symbols from the input. As each a is read, push a 1 onto the stack. As soon as bs are seen, pop a 1 off the stack for each b read. If we reach the end of the string and the stack is nonempty, accept. If the stack becomes empty while we are reading bs, or if we find any as once we start reading bs, reject.

a) Draw the PDA that corresponds to the description above.



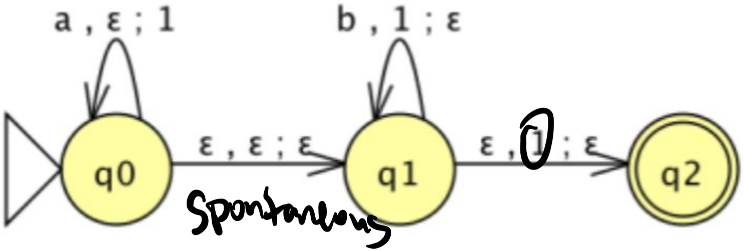
(cf. Sipser Exercises 2.6a/7) Consider the following informal description of a PDA:

> Read symbols from the input. As each a is read, push a 1 onto the stack. As soon as bs are seen, pop a 1 off the stack for each b read. If we reach the end of the string and the stack is nonempty, accept. If the stack becomes empty while we are reading bs, or if we find any as once we start reading bs, reject.

a) Draw the PDA that corresponds to the description above. (Bonus: write the formal definition for the PDA)

nonempty

aaab ✓



a ✓

b ✗

(cf. Sipser Exercises 2.6a/7) Consider the following informal description of a PDA:

> Read symbols from the input. As each a is read, push a 1 onto the stack. As soon as bs are seen, pop a 1 off the stack for each b read. If we reach the end of the string and the stack is nonempty, accept. If the stack becomes empty while we are reading bs, or if we find any as once we start reading bs, reject.

a) Draw the PDA that corresponds to the description above.

b) What is the language recognized by the PDA? What strings are accepted and rejected by the PDA?

(cf. Sipser Exercises 2.6a/7) Consider the following informal description of a PDA:

> Read symbols from the input. As each a is read, push a 1 onto the stack. As soon as bs are seen, pop a 1 off the stack for each b read. If we reach the end of the string and the stack is nonempty, accept. If the stack becomes empty while we are reading bs, or if we find any as once we start reading bs, reject.

a) Draw the PDA that corresponds to the description above.

b) What is the language recognized by the PDA?

Informally, the PDA accepts strings of the form a's followed by b's and there are more a's than b's.

In set builder notation, $L = \{a^i b^j \mid 0 \le j < i\}$

a a a b

Notice that the stack does **not** need to be empty in order to accept a string.

Bonus (if time permits)

Can we construct a PDA for the language $L = \{0^n 1^n 2^n \mid n > 0\}$?

Bonus (if time permits)

Can we construct a PDA for the language $L = \{0^n 1^n 2^n \mid n > 0\}$?

No! We only have a single stack so after matching the number of 1s to the number of 0s, we don't memory of the number of 1s and 0s anymore to match the number of 2s

Thanks for coming! :)