Pumping lemma and Push Down Automata

CSE 105 Week 4 Discussion

Deadlines and Logistics

- No HW this week !
- Schedule your tests asap on PrairieTest !
- Do review quizzes on PrairieLearn
- Review grades for HW 2

Non Regular languages & Pumping Lemma

Non Regular Languages

- Observation 1 : A DFA can only see so far in the past. How far ?
- Automata can only "remember"...
  - ...finitely far in the past
  - ...finitely much information
- If a computation path visits the same state more than once, the machine can't tell the difference between the first time and future times it visited that state.

The pumping lemma

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Understand how this lemma works !

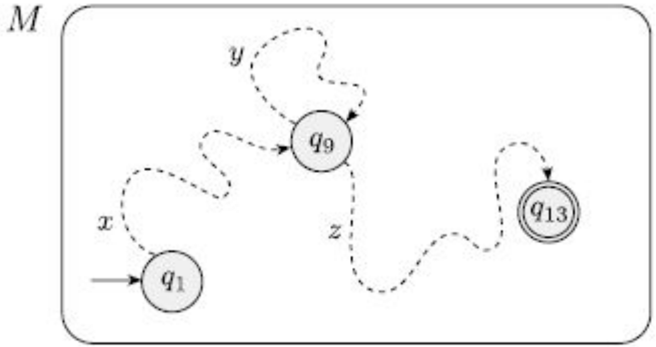
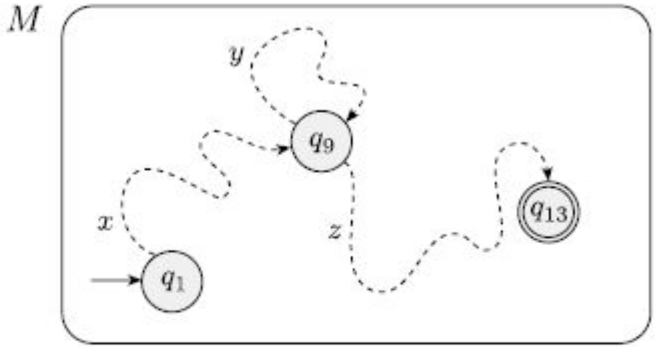


Fig 1.72, Sipser textbook

Understand how this lemma works !



s is any string in A of length at least p

$$s = xyz$$

1. for each $i \geq 0$, $xy^iz \in A$,

2. $|y| > 0$, and

3. $|xy| \leq p$.

Fig 1.72, Sipser textbook

Key points to note

1. Every regular language can be recognized by some DFA
2. Every regular language has a pumping length P.

Statement A : language *L* is regular
Statement B : language *L* has a pumping length P

Pumping lemma states : A⟶B

Note that you cannot conclude that B⟶A ! i.e, just because a language has a pumping length P, it doesn't mean that it is regular !

What are the necessary and sufficient conditions for a language to be regular ?

How to use pumping lemma ?

Recollect that $A \longrightarrow B \equiv \neg B \longrightarrow \neg A$ (CSE 20?) !*

What does this tell us ?

“If a language does NOT have a pumping length, then it is definitely not regular” !

*Contrapositive of an implication is equivalent to the implication itself

Strategy for proving non-regularity

To prove that a language L is not regular:

1. Consider arbitrary positive integer p
2. Prove that p isn't a pumping length for L (adhering to all conditions)
3. Conclude that L does not have any pumping length and is therefore not regular.

Strategy cont.

A positive integer P is the pumping length of a language L if :

$$\forall s \left((|s| \geq p \wedge s \in L) \rightarrow \exists x \exists y \exists z \left(s = xyz \wedge |y| > 0 \wedge |xy| \leq p \wedge \forall i (xy^i z \in L) \right) \right)$$

The negation, “ A positive integer P is NOT the pumping length of a language L if : ”

$$\exists s \left(|s| \geq p \wedge s \in L \wedge \forall x \forall y \forall z \left((s = xyz \wedge |y| > 0 \wedge |xy| \leq p) \rightarrow \exists i (xy^i z \notin L) \right) \right)$$

Strategy cont.

A positive integer P is the pumping length of a language L if :

$$\forall s\left(\left(|s| \geq p \wedge s \in L\right) \rightarrow \exists x \exists y \exists z\left(s = xyz \wedge |y| > 0 \wedge |xy| \leq p \wedge \forall i(xy^i z \in L)\right)\right)$$

The negation, “ A positive integer P is NOT the pumping length of a language L if : ”

$$\exists s\left(|s| \geq p \wedge s \in L \wedge \forall x \forall y \forall z\left(\left(s = xyz \wedge |y| > 0 \wedge |xy| \leq p\right) \rightarrow \exists i(xy^i z \notin L)\right)\right)$$

Although negating the 1st implication to get the 2nd is not part of CSE 105, I urge you to practice the negation !
Understanding first order predicate logic is a very useful skill to have !

$$\exists s \left(|s| \geq p \land s \in L \land \forall x \forall y \forall z \left((s = xyz \land |y| > 0 \land |xy| \leq p) \rightarrow \exists i (xy^i z \notin L)\right)\right)$$

For some P,

There is some string s in the language *L* such that

For all “valid” splits of *s* into *x, y, z*

Repeating *y* i times, for some integer value i throws the resulting string out of the language.

$$\exists s \left(|s| \geq p \wedge s \in L \wedge \forall x \forall y \forall z \left((s = xyz \wedge |y| > 0 \wedge |xy| \leq p) \rightarrow \exists i (xy^i z \notin L) \right) \right)$$

For some P,

-Set P to be an arbitrary integer

There is some string s in the language *L* such that

-Choose s creatively (critical step)

For all “valid” splits of *s* into *x, y, z*

-Define x, y, z according to conditions $|y|>0 \wedge |xy| \leq P$

Repeating *y* i times, for some integer value i throws the resulting string out of the language.

-Choose i such that xy^iz is ejected from L

Is L = $\{a^m b^n \mid 0 \le m \le n\}$ non-regular ?

Is $L = \{a^m b^n \mid m > n \geq 0\}$ Non-regular ?

Which of the following cannot be used for proving that L is non-regular ?

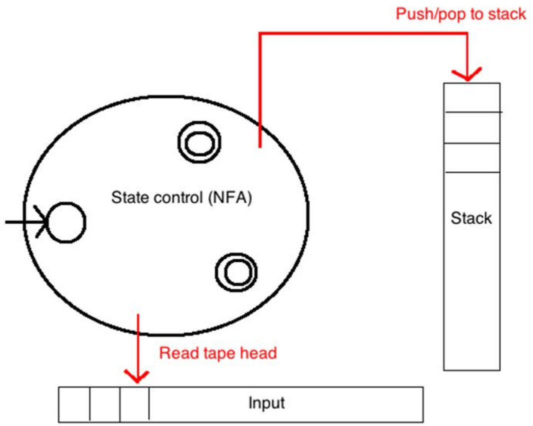
1. $S = aaa$
2. $S = a^P b^P$
3. $S = a^P b^{\lfloor P/2 \rfloor}$
4. $S = a^P b^{P-1}$

Bonus: What i value to choose for the cases that can be used to prove non-regularity ? In other words, prove that L is non regular.

Push-Down Automata

Push-Down Automata

- NFA + Stack for (more) powerful computations



Witnessing acceptance

1. You read the entire input string
2. At least one of computation on the string ends in an accepting state
3. The stack is empty

Witnessing acceptance

1. You read the entire input string
2. At least one of computation on the string ends in an accepting state
3. ~~The stack is empty~~

The stack contents do not directly determine the acceptance of the input string !

Edge label notation (when a, b, c are characters)

- Label $a\ ,\ b\ ;\ c$ or $a\ ,\ b \longrightarrow c$ means
  - Read an a from the input
  - Pop b from the stack
  - Push c to the stack

- Label a , b ; c or a , b ⟶ c means
  - Read an a from the input
  - Pop b from the stack
  - Push c to the stack

What edge label would indicate "Read a 0, don't pop anything from stack, don’t push anything to the stack"?

A. 0, ε → ε
B. ε, 0 → ε
C. ε, ε → 0
D. ε → ε, 0
E. I don't know.

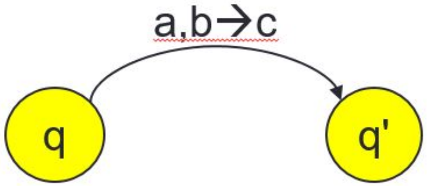
Review the formal definition of a PDA

DEFINITION 2.13

A ***pushdown automaton*** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and 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. $\delta \colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

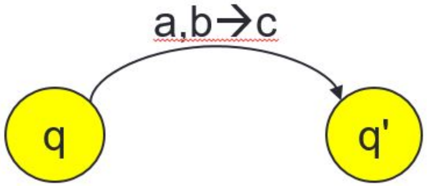
Pop quiz



What would the label ε,ε⟶c mean?

A. If reading the empty string from q and the stack is empty, push c to the stack and move to state q'.
B. If in state q and have finished reading the input string, push c to the stack (without popping any symbol off) and move to state q'

Pop quiz

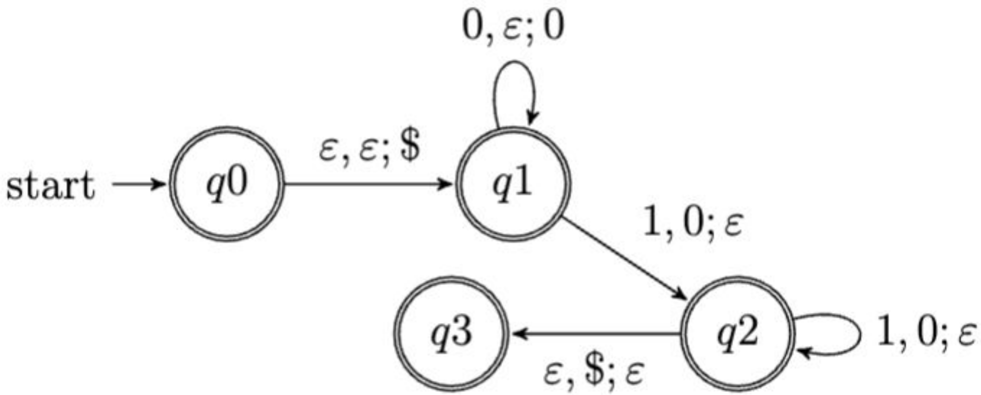


What would the label ε,ε⟶c mean?

A. If reading the empty string from q and the stack is empty, push c to the stack and move to state q'.
B. If in state q and have finished reading the input string, push c to the stack (without popping any symbol off) and move to state q'

NEITHER ! "look at the stack, and irrespective of its contents, transition from q to q' without consuming a symbol and then push a c to the stack"

Review quiz



Select all and only the strings below that are accepted by this PDA.

- [ ] 111
- [ ] 11
- [ ] 00
- [ ] 000

Select all possible options that apply.