Week 4 at a glance

Textbook reading: Section 1.4, 2.2, 2.1.

Before Monday, read Introduction to Section 1.4 (page 77) which introduces nonregularity.

Before Wednesday, read Definition 2.13 (page 111-112) introducing Pushdown Automata.

Before Friday, read Example 2.18 (page 114).

For Week 5 Monday: read Introduction to Section 2.1 (pages 101-102).

We will be learning and practicing to:

- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
  - Give examples of sets that are regular (and prove that they are).
    - **State the definition of the class of regular languages**
    - **Explain the limits of the class of regular languages**
    - **Identify some regular sets and some nonregular sets**
  - Use precise notation to formally define the state diagram of a PDA
  - Use clear English to describe computations of PDA informally.
    - **Define push-down automata informally and formally**
    - **State the formal definition of a PDA**
    - **Trace the computation(s) of a PDA on a given string using its state diagram**
    - **Determine if a given string is in the language recognized by a PDA**
    - **Translate between a state diagram and a formal definition of a PDA**
    - **Determine the language recognized by a given PDA**
- Know, select and apply appropriate computing knowledge and problem-solving techniques.
  - Apply classical techniques including pumping lemma, determinization, diagonalization, and reduction to analyze the complexity of languages and problems.
    - **Justify why the Pumping Lemma is true.**
    - **Use the pumping lemma to prove that a given language is not regular.**

TODO:

Schedule your Test 1 Attempt 1, Test 2 Attempt 1, Test 1 Attempt 2, and Test 2 Attempt 2 times at PrairieTest (http://us.prairietest.com)

Review Quiz 3 on PrairieLearn (http://us.prairielearn.com), due 1/29/2025

Homework 2 submitted via Gradescope (https://www.gradescope.com/), due 1/30/2025

Review Quiz 4 on PrairieLearn (http://us.prairielearn.com), due 2/5/2025

CC BY-NC-SA 2.0 Version February 9, 2025 (1)

Monday: Pumping Lemma

Definition and Theorem: For an alphabet Σ, a language L over Σ is called **regular** exactly when L is recognized by some DFA, which happens exactly when L is recognized by some NFA, and happens exactly when L is described by some regular expression

We saw that: The class of regular languages is closed under complementation, union, intersection, set-wise concatenation, and Kleene star.

Extra practice:

Disprove: There is some alphabet Σ for which there is some language recognized by an NFA but not by any DFA.

Disprove: There is some alphabet Σ for which there is some finite language not described by any regular expression over Σ.

Disprove: If a language is recognized by an NFA then the complement of this language is not recognized by any DFA.

Fix alphabet Σ. Is every language L over Σ regular?

Set	Cardinality
$\{0,1\}$	
$\{0,1\}^*$	
$\mathcal{P}(\{0,1\})$	
The set of all languages over $\{0,1\}$	
The set of all regular expressions over $\{0,1\}$	
The set of all regular languages over $\{0,1\}$	

CC BY-NC-SA 2.0 Version February 9, 2025 (2)

Strategy: Find an **invariant** property that is true of all regular languages. When analyzing a given language, if the invariant is not true about it, then the language is not regular.

Pumping Lemma (Sipser Theorem 1.70): If A is a regular language, then there is a number p (a *pumping length*) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$ such that

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

Proof idea: In DFA, the only memory available is in the states. Automata can only "remember" finitely far in the past and finitely much information, because they can have only finitely many states. If a computation path of a DFA visits the same state more than once, the machine can't tell the difference between the first time and future times it visits this state. Thus, if a DFA accepts one long string, then it must accept (infinitely) many similar strings.

Proof illustration

CC BY-NC-SA 2.0 Version February 9, 2025 (3)

True or False: A pumping length for $A = \{0,1\}^*$ is $p = 5$.

True or False: A pumping length for $A = \{0,1\}^*$ is $p = 2$.

True or False: A pumping length for $A = \{0,1\}^*$ is $p = 105$.

Restating **Pumping Lemma**: If L is a regular language, then it has a pumping length.

Contrapositive: If L has no pumping length, then it is nonregular.

The Pumping Lemma *cannot* be used to prove that a language *is* regular.

The Pumping Lemma **can** be used to prove that a language *is not* regular.

Extra practice: Exercise 1.49 in the book.

Proof strategy: To prove that a language L is **not** regular,

- Consider an arbitrary positive integer p
- Prove that p is not a pumping length for L
- Conclude that L does not have *any* pumping length, and therefore it is not regular.

Negation: A positive integer p is **not a pumping length** of a language L over Σ iff

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

CC BY-NC-SA 2.0 Version February 9, 2025 (4)

Wednesday: Proving nonregularity, and beyond

Proof strategy: To prove that a language L is **not** regular,

- Consider an arbitrary positive integer p
- Prove that p is not a pumping length for L. A positive integer p is **not a pumping length** of a language L over Σ iff

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

  Informally:

- Conclude that L does not have *any* pumping length, and therefore it is not regular.

Example: $\Sigma = \{0, 1\}$, $L = \{0^n 1^n \mid n \geq 0\}$.

Fix p an arbitrary positive integer. List strings that are in L and have length greater than or equal to p:

Pick $s =$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$.

Then when $i =$, $xy^i z =$

CC BY-NC-SA 2.0 Version February 9, 2025 (5)

Example: $\Sigma = \{0, 1\}$, $L = \{ww^{\mathcal{R}} \mid w \in \{0, 1\}^*\}$. Remember that the reverse of a string w is denoted $w^{\mathcal{R}}$ and means to write w in the opposite order, if $w = w_1 \cdots w_n$ then $w^{\mathcal{R}} = w_n \cdots w_1$. Note: $\varepsilon^{\mathcal{R}} = \varepsilon$.

Fix p an arbitrary positive integer. List strings that are in L and have length greater than or equal to p:

Pick $s =$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$.

Then when $i =$, $xy^iz =$

Example: $\Sigma = \{0, 1\}$, $L = \{0^j1^k \mid j \geq k \geq 0\}$.

Fix p an arbitrary positive integer. List strings that are in L and have length greater than or equal to p:

Pick $s =$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$.

Then when $i =$, $xy^iz =$

Example: $\Sigma = \{0, 1\}$, $L = \{0^n1^m0^n \mid m, n \geq 0\}$.

Fix p an arbitrary positive integer. List strings that are in L and have length greater than or equal to p:

Pick $s =$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$.

Then when $i =$, $xy^iz =$

CC BY-NC-SA 2.0 Version February 9, 2025 (6)

Extra practice:

Language	$s \in L$	$s \notin L$	Is the language regular or nonregular?
$\{a^n b^n \mid 0 \leq n \leq 5\}$			
$\{b^n a^n \mid n \geq 2\}$			
$\{a^m b^n \mid 0 \leq m \leq n\}$			
$\{a^m b^n \mid m \geq n+3, n \geq 0\}$			
$\{b^m a^n \mid m \geq 1, n \geq 3\}$			
$\{w \in \{a,b\}^* \mid w = w^{\mathcal{R}}\}$			
$\{ww^{\mathcal{R}} \mid w \in \{a,b\}^*\}$			

CC BY-NC-SA 2.0 Version February 9, 2025 (7)

Friday: Pushdown Automata

Regular sets are not the end of the story

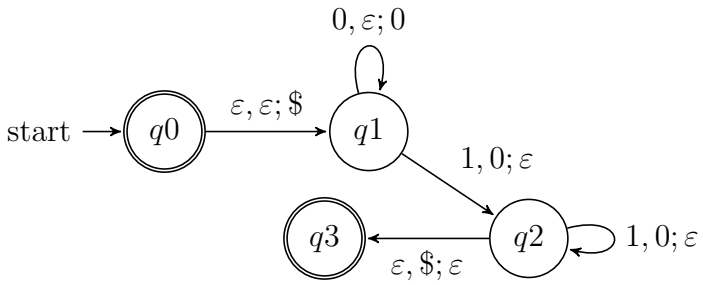
- Many nice / simple / important sets are not regular
- Limitation of the finite-state automaton model: Can't "count", Can only remember finitely far into the past, Can't backtrack, Must make decisions in "real-time"
- We know actual computers are more powerful than this model...

The **next** model of computation. Idea: allow some memory of unbounded size. How?

- To generalize regular expressions: **context-free grammars**
- To generalize NFA: **Pushdown automata**, which is like an NFA with access to a stack: Number of states is fixed, number of entries in stack is unbounded. At each step (1) Transition to new state based on current state, letter read, and top letter of stack, then (2) (Possibly) push or pop a letter to (or from) top of stack. Accept a string iff there is some sequence of states and some sequence of stack contents which helps the PDA processes the entire input string and ends in an accepting state.

Is there a PDA that recognizes the nonregular language $\{0^n 1^n \mid n \geq 0\}$?

CC BY-NC-SA 2.0 Version February 9, 2025 (8)



The PDA with state diagram above can be informally described as:

> Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If the stack becomes empty and we are at the end of the input string, accept the input. If the stack becomes empty and there are 1s left to read, or if 1s are finished while the stack still contains 0s, or if any 0s appear in the string following 1s, reject the input.

Trace a computation of this PDA on the input string 01.

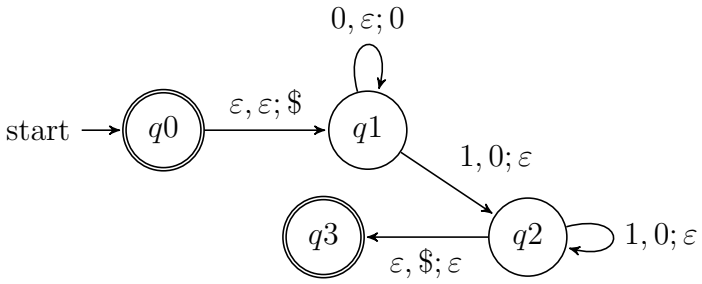
Extra practice: Trace the computations of this PDA on the input string 011.

CC BY-NC-SA 2.0 Version February 9, 2025 (9)

A PDA recognizing the set { } can be informally described as:

> Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If the stack becomes empty and there is exactly one 1 left to read, read that 1 and accept the input. If the stack becomes empty and there are either zero or more than one 1s left to read, or if the 1s are finished while the stack still contains 0s, or if any 0s appear in the input following 1s, reject the input.

Modify the state diagram below to get a PDA that implements this description:



CC BY-NC-SA 2.0 Version February 9, 2025 (10)