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

Version January 27, 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.

Let Σ be arbitrary alphabet. WTS each language recognized by an NFA over Σ is recognized by some DFA over Σ. Use Macro State construction.

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

Let Σ be arbitrary language. By induction on size of language: if language has size 0 then it is $\emptyset$ and is described by $\emptyset$. Let n be arbitrary nonneg int. Suppose as IH that each language of size n can be described by some regular expression. Let A be language of size $n+1$. Let w_0 be the string or least element of A. $A = A \setminus \{w_0\} \cup \{w_0\}$ and by IH there is a regular expression R_n with $L(R_n) = A \setminus \{w_0\}$ because $|A \setminus \{w_0\}| = n$.
Also, by definition of regular expressions w_0 is itself a regular expression with $L(w_0) = \{w_0\}$. Thus the regular expression $R = R_n \cup w_0$ witnesses the result.

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

For N an NFA, the macro-state construction gives DFA M with $L(M) = L(N)$ and using the "flip status of states" construction we get a DFA M' with $L(M') = \overline{L(M)} = \overline{L(N)}$.

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

Diagonalization:
The power set of a countably infinite set is uncountable

Set	Cardinality
$\{0,1\}$	2
$\{0,1\}^*$	countably infinite.
$\mathcal{P}(\{0,1\})$	
The set of all languages over $\{0,1\}$ $\mathcal{P}(\{0,1\}^*)$	uncountable
The set of all regular expressions over $\{0,1\}$ $0 \quad 1 \quad \cup \quad \circ \quad * \quad (\quad)$	countably infinite
The set of all regular languages over $\{0,1\}$	countably infinite

$\mathcal{P}(\Sigma^*)$

· Non reg · Non reg · Regular languages · Non reg

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.

DFA

n states

accepts string w, $|w| \geq n$ $q_0, -, q_0, -, \cdot q_0 \cdots, -$ $|w|+1$ many slots

$$xz$$
$$xyyz$$
$$xyyyz$$

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 $i \geq 0$, $xy^i z \in A$
- $|xy| \leq p$.

Regular languages have some nice structure

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   Given regular language A, let M be DFA with $L(M) = A$. Consider p being the number of states in M. For any string of length $\geq p$ in $L(M)$ there must be a repeated state in the computation of M on the string.

A pos int P is a pumping length for language L means for
each $s \in L$ with $|s| \geq p$ $\exists x \exists y \exists z (s=xyz \land |xy| \leq p \land |y| > 0 \land$
$\forall i (xy^i z \in L))$

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

Consider arbitrary $s \in A$ with $|s| \geq 5$.
Take $x = \varepsilon$, $y = s_1$ (the first character in s), z is
the rest of s (from the second character on). Notice $xy^i z \in \{0,1\}^*$
for all $i \geq 0$. A.

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

Similar

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

Similar

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

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

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:

There is a long string in the language so that no matter how we choose x, y, z so $s = xyz$ and $|y| > 0$ and $|xy| \leq p$ there will be some number i so that $xy^i z \notin L$

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

$$01 \quad , \quad 0^p 1^p \quad , \quad \varepsilon \quad \text{in } L$$
$$11 \quad , \quad 10 \quad \text{not in } L$$

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:

$$0^{2p} 1^{2p} \qquad \overline{0^{p-1} 1^{p-1}}$$

Pick $s = 0^p 1^p$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$. There are ints k, r so that

$$00 \overbrace{| 0 \cdots | 0 1}^{p} \cdots 111$$

$x = 0^k$

$y = 0^r, r > 0$

$z = 0^{p-k-r} 1^p$

Then when $i = 0$, $xy^i z = \underbrace{0^k 0^{p-k-r}}_{p-r \neq p} 1^p \notin L$ because $r > 0$

$\cancel{1}$ $\quad xy^i z = 0^k \cancel{0^r} \underbrace{0^{p-k-r}}_{= p} 1^p = s \in L$

$2 \quad xy^2 z = 0^k \underbrace{0^r 0^r 0^{p-k-r}}_{r+p \neq p} 1^p \notin L$ because $r > 0$

0110 , 111111 in L

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:

$0^p 11 0^p$, $0^p 1^p 1^p 0^p$, $0^p 0^p$

Pick $s = 0^p 11 0^p$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$. There are nonneg ints k, r so that

$x = 0^k$ $y = 0^r$ $z = 0^{p-k-r} 11 0^p$

$$ \underbrace{00 \cdots \cdots 0}_{p} \ 11 \ \underbrace{0 \cdots\cdots 00}_{p} $$

Then when $i = 0$, $xy^i z = 0^k 0^{p-k-r} 11 0^p = 0^{p-r} 11 0^p$ which is not an even length palindrome because either r is odd so $|xy^i z|$ is odd or r is even and since $r > 0$, $r \geq 2$, so $\frac{p-r+2+p}{2} = \frac{2p+2-r}{2} < p+1$ which means the second half of the string is all 0s so can't be the reverse of the first half of the string, which has 1s.

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

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

ε , 0 , 01 , 0001 ... in L

Pick $s = 0^p 1^p$

num of 0s to be greater than or equal to num of 1s

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$. There are nonneg ints k, r where

$$ \underbrace{00 \cdots \cdots 01}_{p} \ \underbrace{\cdots\cdots 11}_{p} $$

$x = 0^k$
$y = 0^r$
$z = 0^{p-k-r} 1^p$

Then when $i = 0$, $xy^i z = 0^k 0^{p-k-r} 1^p$ $\notin L$

because $k + p - k - r = p - r < p$.

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

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

Pick $s = 0^p 1 0^p$

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$. There are nonneg ints k, r so that
$x = 0^k$ $y = 0^r$ $z = 0^{p-r} 1 0^p$.

$$ \underbrace{00 \cdots\cdots 010}_{p} \ \underbrace{\cdots\cdots 00}_{p} $$

Then when $i = 2$, $xy^i z = 0^k 0^r 0^r 0^{p-r} 1 0^p = 0^{p+r} 1 0^p$ $\notin L$ because the number of 0s at the start of the string doesn't match the number of 0s at the end.

Version January 27, 2025 (6)

Extra practice:

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

(we recommend you try these out and then check your work with us in office hours)

Friday: Pushdown Automata

$$\{0^n 1^n \mid n \geq 0\} \quad \text{nonregular}$$

$$\{0^n 1^m \mid n, m \geq 0\} = L\left(0^* \cdot 1^*\right)$$

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?

Sec 2.1
- To generalize regular expressions: **context-free grammars**   syntactic

computation model

Sec 2.2
- 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.

input alphabet Σ

state diagram
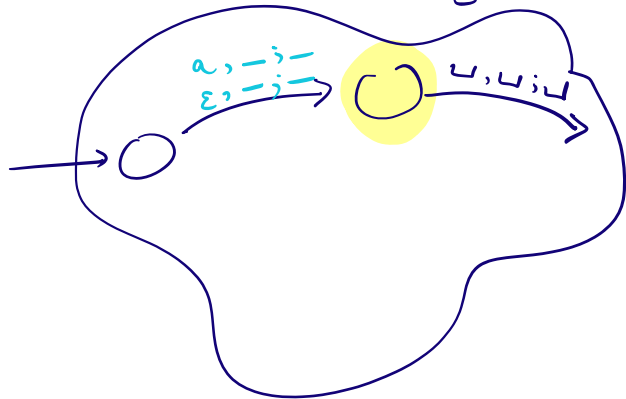
Stack : first in   last out
last in   first out

$a, -;-$
$\varepsilon, -;-$

$\sqcup, \sqcup; \sqcup$

characters from stack alphabet

$a \in \Sigma$
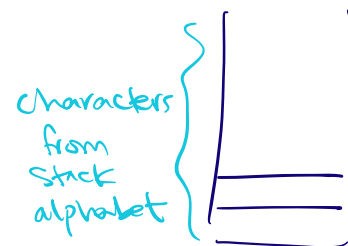
arrow label :   input char to read   , top char of stack   when I follow
                  or ε to pop from stack;   this arrow
                                            or ε what char to
Is there a PDA that recognizes the nonregular language $\{0^n 1^n \mid n \geq 0\}$?   push onto top
                                                                                        of stack
                                                                                          or ε

Idea: As read string, start by
looking for 0s   (if see 1 before 0, reject)
and push each 0 to the stack and
when I see 1s, match them to 0s in
the stack. Make sure stack is empty
when I finish reading the string.

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

Annotations on the diagram:
- without consuming input
- without needing any char to be at top of stack
- $0, \varepsilon; 0$ — as read 0s, push 0s to stack
- $\varepsilon, \varepsilon; \$$
- start → $q0$ — accept the empty string
- $q1$
- $1, 0; \varepsilon$ — pop 0 from stack for first 1 read
- input alphabet $\{0, 1\}$
- stack alphabet $\{\$, 0\}$
- used to detect emptiness
- $\boxed{\$}$
- $q3$
- $\varepsilon, \$; \varepsilon$ — if there's no more 0 in stack can move to accepting state
- $q2$
- $1, 0; \varepsilon$ — pop 0 from stack for each 1 read

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 (accepting) of this PDA on the input string 01.

q_0 , q_1 , q_1 , q_2 , q_3 ACC.

stack: $\sqcup$, $\boxed{\$}$, $\boxed{\begin{matrix}0\\\$\end{matrix}}$, $\boxed{\$}$, $\sqcup$

input: $\downarrow$ 01 , $\downarrow$ 01 , 0$\downarrow$1 , 01$\downarrow$, 01$\downarrow$

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

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: