CSC236 Week 9

Larry Zhang

NEW TOPIC

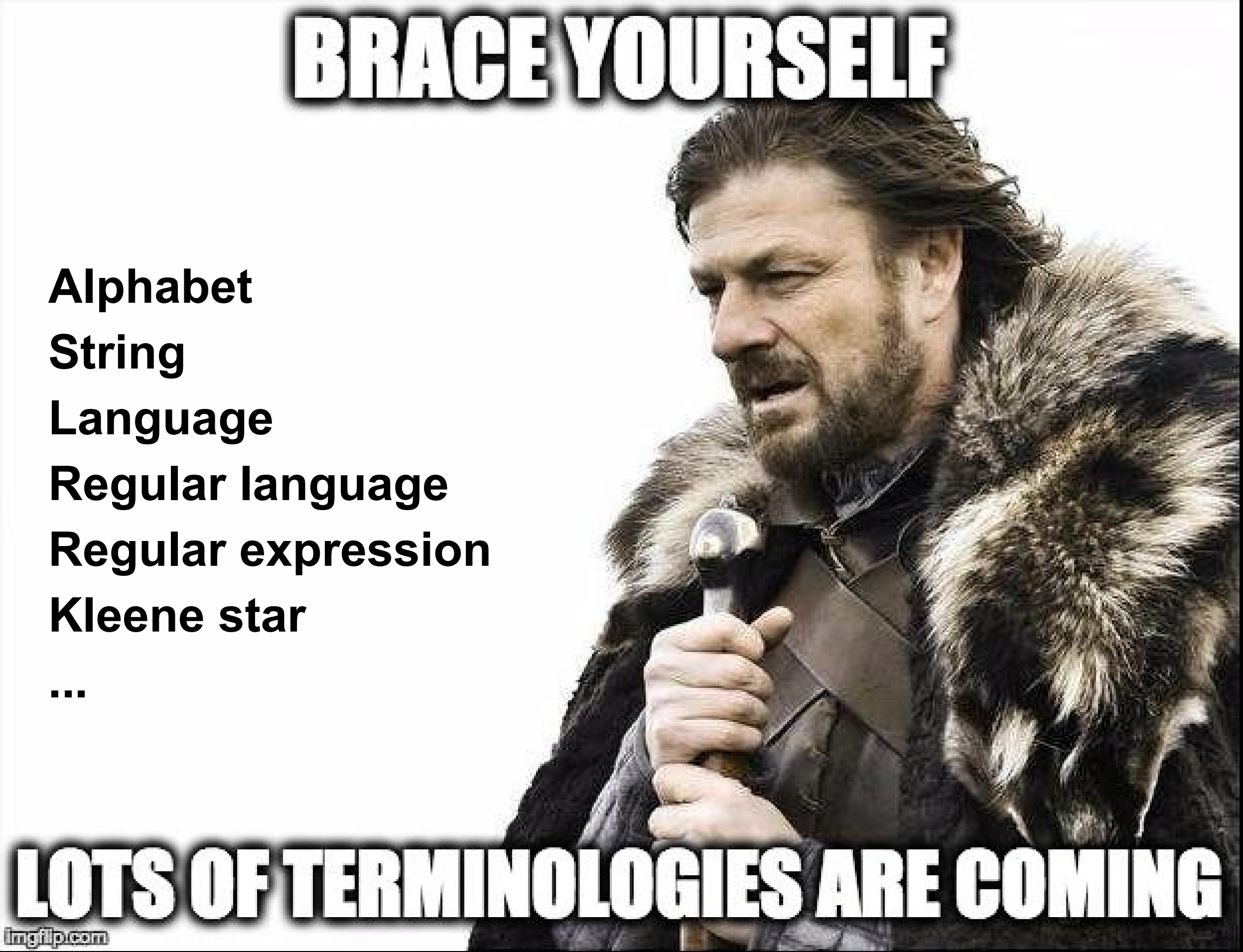
Finite Automata & Regular Language

Finite Automata

- An important part of the theory of computation
- A simple and powerful model for computation
  - It describes a simple idealized machine (the theoretical computer)
- It has many applications
  - Digital circuit design (you'll use it in CSC258)
  - Compiler and interpreter (how computer understands your program)
  - Text searching/parsing, lexical analysis, pattern matching
  - Neural networks (models the working mechanism of the neurons)
  - etc.
- Yet another hallmark for a CS pro.

The application we focus on in CSC236

Regular Language & Regular Expressions


BRACE YOURSELF
Alphabet
String
Language
Regular language
Regular expression
Kleene star
...
LOTS OF TERMINOLOGIES ARE COMING
imgflip.com

Terminology: **Alphabet**

- Alphabet: a finite set of symbols
  - e.g., {0, 1} # the alphabet for binary strings
  - e.g., {a, b, c, …, z} # the alphabet for English words
  - etc
- We denote an alphabet using the Greek letter ...

Σ (“sigma”)

Has nothing to do with summation.

Terminology: **String**

- A string ***w*** over alphabet ∑ is a finite sequence of **symbols from** ∑
- For example, the following are some strings over the alphabet {0, 1}
  - "0", "1", "0110", "1110"
- Each English word is a string over the alphabet {a, b, …, z}
  - e.g., "brace", "yourself"
- A special string: the **empty string** "" which we denote with **ε**
  - It is a string over **any** alphabet.

Terminology: **Length** of a String

- The **length** of string ***w*** is the number of characters in ***w***
- It is written as $|w|$, for example
  - $|\text{brace}| = 5$
  - $|010111| = 6$
  - $|\varepsilon| = 0$

- Σ^n: set of all strings over alphabet Σ of length n
- Σ^*: set of all strings over alphabet Σ

Exercises

Consider alphabet $\sum = \{0, 1\}$

What is Σ^0?

- set of all binary strings with length 0
- A set with one string (the empty string) in it: **{ε}**

► Σ^n: set of all strings over alphabet Σ of length n

Consider alphabet $\sum = \{0, 1\}$

What is Σ^1 ?

- set of all binary strings of length 1
- A set with two strings in it: **{0, 1}**

► Σ^n: set of all strings over alphabet Σ of length n

Consider alphabet $\sum = \{0, 1\}$

What is Σ^2?

- set of all binary strings of length 2
- A set with four strings in it: **{00, 01, 10, 11}**

▸ Σ^n: set of all strings over alphabet Σ of length n

Consider alphabet $\sum = \{0, 1\}$

What is Σ^{12} ?

- set of all binary strings of length 12
- **{000000000000, …, 111111111111}**
- A set with **4096** (2^12) strings in it:

► Σ^n: set of all strings over alphabet Σ of length n

Consider alphabet $\sum = \{0, 1\}$

What is Σ^* ?

- set of all binary strings
- of length from 0 to arbitrarily large
- A set with **an infinite number** of strings in it.

▸ Σ^*: set of all strings over alphabet Σ

Consider alphabet $\sum = \{0, 1\}$

What is $\Sigma^0 \cup \Sigma^1$?

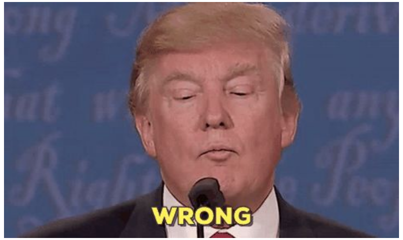
- set of all binary strings of length 0 or 1
- A set with three strings in it: **{ε, 0, 1}**

back to terminologies ...

Warm-Up

Consider the alphabet be $\sum$ **= {a, b, c, …, z}**

then the set of all English words ***L*** is basically $\sum$*



L is only a **subset** of $\sum$*, i.e., ***L*** $\subseteq \sum$*

- Not all strings in $\sum$* are in L, such as “sdfasdf”, “ttttt” and the empty string.

Terminology: **Language**

- A **language *L*** over alphabet ∑ is a subset of ∑*
  - i.e., $L \subseteq \Sigma^*$
  - The set of all English words is a language over ∑ = {a, b,..., z}
- Languages can have finite or infinite size, e.g.,
  - The English language is finite

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

$L_1 = \{\epsilon, a, b, ccc\}$ **is finite**

$L_2 = \{w \in \{a, b, c\}^* \mid |w| \leq 3\}$ **is finite**

$L_3 = \{w \in \{a, b, c\}^* \mid w \text{ has the same number of a's and c's}\}$ **is infinite**

Operations on Languages

Recursive define set, whose properties can be proven by **structural induction**!

Given two languages **L, M ⊆ ∑***, **three operations** can be used to **generate new languages**.

- **Union**, **L** ∪ **M**: all strings from **L** together with all strings from **M**.

$$L \cup M = \{x \in \Sigma^* \mid x \in L \text{ or } x \in M\}$$

- **Concatenation**, **LM**: concatenate each string in **L** with each string in **M**.

$$LM = \{xy \in \Sigma^* \mid x \in L, y \in M\}$$

- **Kleene Star**, **L***: all strings that can be formed by concatenating zero or more strings from **L**.

$$L^* = \{\epsilon\} \cup \{x \in \Sigma^* \mid \exists w_1, w_2, \ldots, w_n \in L \text{ such that } x = w_1 w_2 \ldots w_n, \text{ for some } n\}$$

Exercises

Kleene star example

Let language L = {0, 1}, then what is L* ?

Answer: the set of all binary strings, including the empty string.

$$L^* = \{\epsilon\} \cup \{x \in \Sigma^* \mid \exists w_1, w_2, \ldots, w_n \in L \text{ such that } x = w_1 w_2 \ldots w_n, \text{ for some } n\}$$

Consider languages **L = {a, aa}** and **M = {a, cc}**

What is **L** ∪ **M**?

L ∪ **M = {a, aa, cc}**

$$L \cup M = \{x \in \Sigma^* \mid x \in L \text{ or } x \in M\}$$

Consider languages **L = {a, aa}** and **M = {a, cc}**

What is **LM**?

LM = {a.a, a.cc, aa.a, aa.cc}

(the **dot** is for visualizing concatenation, it does NOT count as a character)

$$LM = \{xy \in \Sigma^* \mid x \in L, y \in M\}$$

Consider languages **L = {a, aa}** and **M = {a, cc}**

What is **L***?

L* = {ε, a, aa, aaa, aaaa, aaaaa, …}

better description:

L* = {w | w consists of 0 or more a's}

$L^* = \{\epsilon\} \cup \{x \in \Sigma^* \mid \exists w_1, w_2, \ldots, w_n \in L \text{ such that } x = w_1 w_2 \ldots w_n, \text{ for some } n\}$

Consider languages **L = {a, aa}** and **M = {a, cc}**

What is **M***?

M* = {ε, a, aa, cc, acc, cca, aaa, aacc, ccaa, ~~ccc~~, cccc, …}

M* = {w | w consists of 0 or more a’s and c’s, and all c’s are in pairs }

M* = {w | w consists of 0 or more a’s and cc’s}

$$L^* = \{\epsilon\} \cup \{x \in \Sigma^* \mid \exists w_1, w_2, \ldots, w_n \in L \text{ such that } x = w_1 w_2 \ldots w_n, \text{ for some } n\}$$

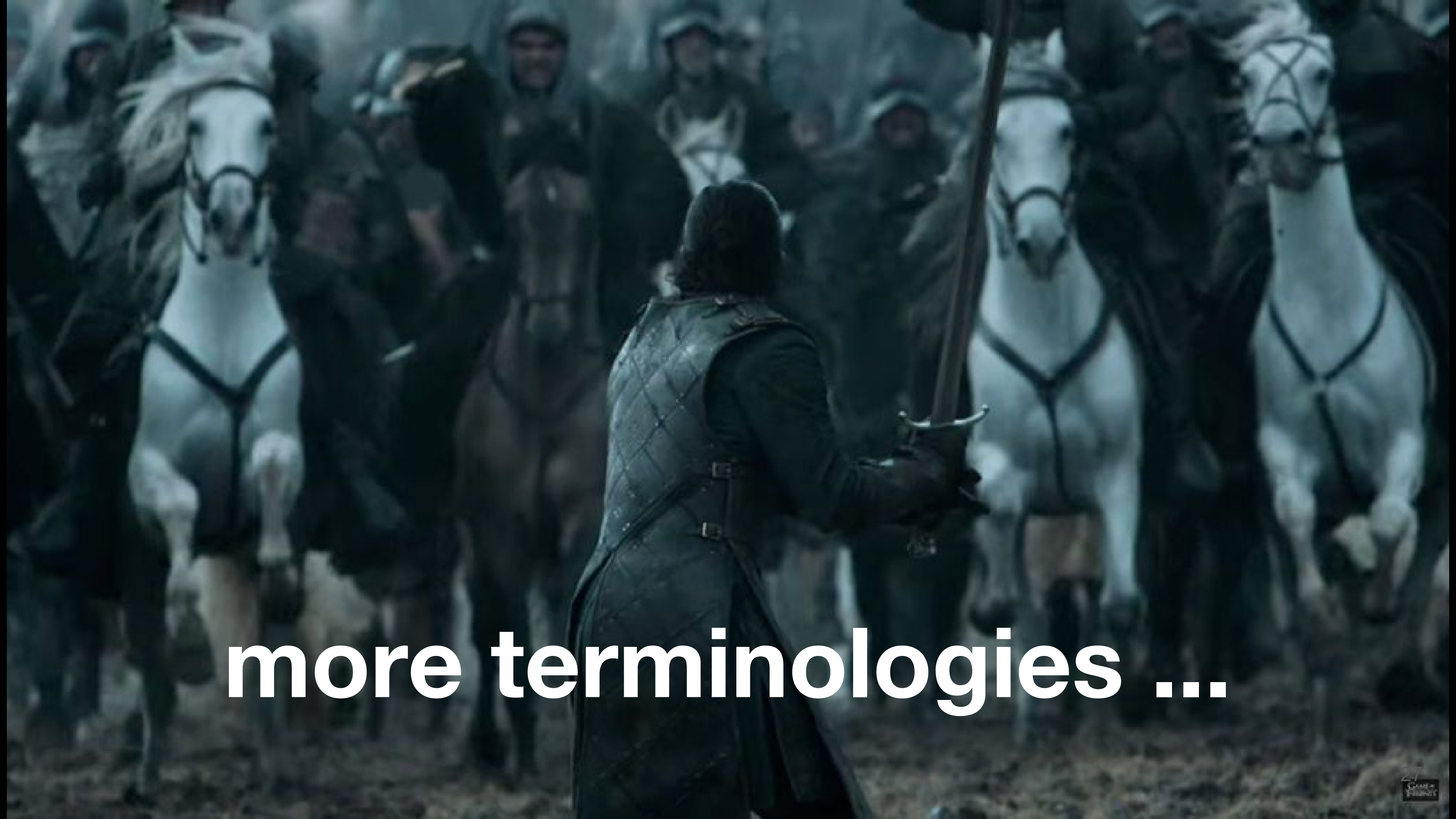
Consider languages **L = {a, aa}** and **M = {a, cc}**

What is (**L** ∪ **M**)*?

(L ∪ M)* = {a, aa, cc}*

(L ∪ M)* = {w | w consists of 0 or more a’s and c’s, and all c’s are in pairs }

It is the same set as M*



more terminologies ...

Terminology: Regular Languages

Regular languages are a **subset** of all languages. (Not all languages are regular)

The **set of regular languages** over alphabet ∑ is **recursively** defined as follows

- ∅, the empty set, is a regular language
- **{ε}**, the language consisting of only the empty string, is a regular language
- For any symbol **a** ∈ ∑, **{a}** is a regular language.
- If **L**, **M** are regular languages, then so are **L** ∪ **M**, **LM**, and **L***.

Quick Exercise

- ∅, the empty set, is a regular language
- **{ε}**, the language consisting of only the empty string, is a regular language
- For any symbol **a** ∈ ∑, **{a}** is a regular language.
- If L, M are regular languages, then so are **L** ∪ **M**, **LM**, **L*** and **M***.

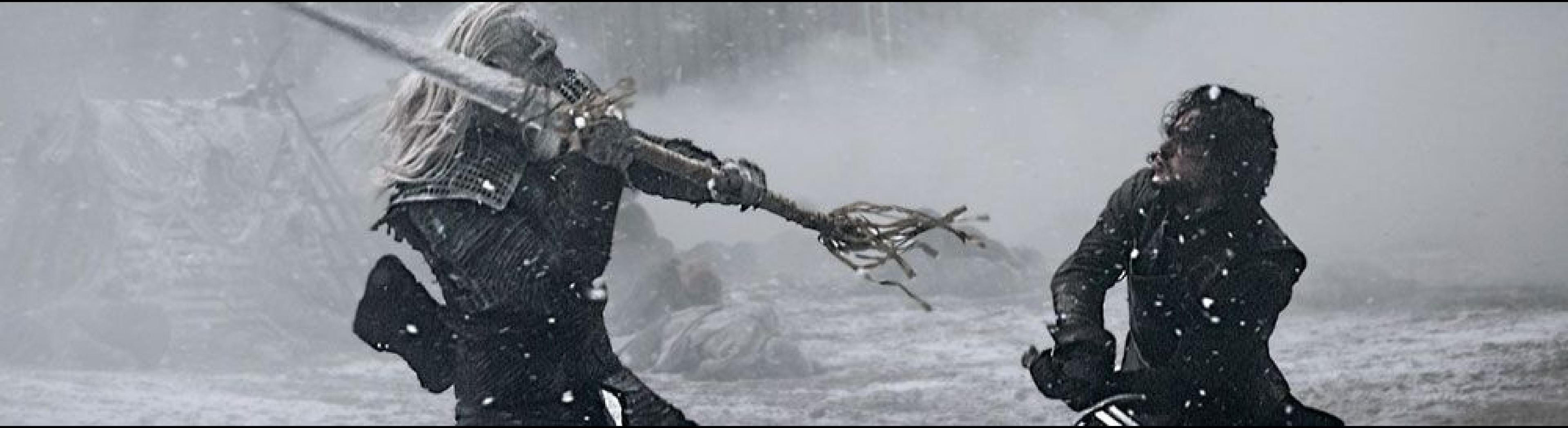
Prove that language **L = {a, aa}** is a regular language.

Proof:

{a} is regular by definition

So {aa} = {a}{a} is regular (concatenation rule)

So {a, aa} = {a} U {aa} is regular (union rule)



one last terminology ...

Terminology: **Regular Expressions**

A **regular expression** (regex) is a **string representation** of a regular language.

A regex "**matches**" a set of strings (the represented regular language).

It also has a recursive definition:

For a regex r, $\mathcal{L}(r)$ is the language matched by r

- $\emptyset$ is a regex, with $\mathcal{L}(\emptyset) = \emptyset$ (matches no string)
- ϵ is a regex, with $\mathcal{L}(\epsilon) = \{\epsilon\}$ (matches only the empty string)
- For all symbols $a \in \Sigma$, a is a regex with $\mathcal{L}(a) = \{a\}$

these are the bases, and there is more ...

Definition of regex, continued ...

- Let r, r_1, and r_2 be regular expressions
  - $r_1 + r_2$ is a regex, with $\mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$
  - $r_1 r_2$ is a regex, with $\mathcal{L}(r_1 r_2) = \mathcal{L}(r_1)\mathcal{L}(r_2)$
  - r^* is a regex, with $\mathcal{L}(r^*) = (\mathcal{L}(r))^*$

More Exercises

EX1: Find Language from Regular Expression

Describe the language represented by the following regular expression.

r: $(01)+1(0+1)^*$

The * has the highest precedence.

It matches the string 01

Must start with 1,
followed by any string with 0's or 1's

L(r): set with 01, and all strings with 0's and 1's that start with 1

EX2: Develop Regular Expression from Language

Give a regular expression that represents the following language.

$$L = \{w \in \{a, b\}^* \mid |w| \leq 2\}$$

Naive way: enumerate all possible strings

- ε + a + aa + b + bb + ab + ba

Then try to combine some of them

- ε + a(ε + a) + b(ε + b) + ab + ba
- or, **(ε + a + b)(ε + a + b)**

Let's try to expand (ε + a + b)(ε + a + b)

= εε + εa + εb + aε + aa + ab + bε + ba + bb
= ε + a + b + aa + ab + ba + bb

\# εε = ε
\# εa = aε = a
\# εb = bε = b

EX3

Give a regular expression that represents the following language

$$L = \{w \in \{0,1\}^* \mid w \text{ does not have } 11 \text{ as a substring}\}$$

Naive way: enumerate all possible strings

- No can do! There are infinitely many strings in L.

What should match:

- ε, 0, 1, 00, 01, 10, 100, 101, 001, 01010010100

What should NOT match:

- 11, 111, 011, 110, 1111, 01110, 00000011

$$L = \{w \in \{0,1\}^* \mid w \text{ does not have 11 as a substring}\}$$

EX3

- **Observation 1**: if we have a 1, what must happen after that 1?
  - There must be a 0 or nothing.
- **Observation 2**: 0’s can go anywhere and there can be as many of them as you want.

$L = \{w \in \{0,1\}^* \mid w \text{ does not have 11 as a substring}\}$

EX3

Attempt #1: **(10)***

- It matches ε, 10, 1010, 101010, …, all in L.
- But it does NOT match everything needed.
- It's missing strings with multiple 0's in a row
  - Like, 100, 10010000

Try to fix this: **(100*)***

$L = \{w \in \{0,1\}^* \mid w \text{ does not have 11 as a substring}\}$

EX3

Attempt #2: **(100*)***

- Now I'm allowing multiple 0's in a row. Right?
- It's still missing some strings …
  - 01, 001, 0001010, 000000101000
  - The strings that start with 0's.

Try to fix this: **0*(100*)***

$L = \{w \in \{0,1\}^* \mid w \text{ does not have } 11 \text{ as a substring}\}$

EX3

Attempt #3: 0*(100*)*

- Now I’m allowing the string to start with 0 or 1. It’s good, right?
- It’s still missing some strings …
  - 1, 101, 0010101, 00100001…
  - the strings that end with 1’s. How to fix it?

Attempt #4: 0*(100*)*(ε + 1)

- **This one is correct!**

Regex **$0^*(100^*)^*(\varepsilon + 1)$** represents the language

$$L = \{w \in \{0,1\}^* \mid w \text{ does not have 11 as a substring}\}$$

This regex is NOT unique. The following is also a valid regex for the above language.

$$\mathbf{(\varepsilon+1)(00^*1)^*0^*}$$

Verify it yourself!

Home Exercise

N = {w : {0,1}* | w represents a binary number divisible by 2}

Find a regex for language N.

Takeaway

For a regex to correctly represent a language **L**, it must match **every** string in L, and **nothing** else.

The regex is wrong if **any** of the following happens

- There is a string in L, that the regex does not match.
- There is a string that is not in L, but is matched by the regex.

The general steps of coming up with a regex

- Observe and understand the pattern that need to be matched, educatedly attempt a regex
- Verify if the attempted regex is wrong (above two criteria), if wrong, know the reason, fix it.
- Repeat the above until you're convinced you have right answer.

Next week

DFA: model regular expression as a computation.