# CSE 311 Section 09

## Models of Computation

# Administrivia

# Announcements & Reminders

- HW6 Regrade Requests
  - Submit a regrade request if something was graded incorrectly

- HW7
  - Due **this Friday** 11/21 @11:59pm
  - Late due date Monday 11/24 @ 11:59pm

- HW8
  - Due Friday 12/05 @ 11:59pm

- Final Exam
  - Monday 12/08 @ 12:30pm-2:20pm
  - Keep an eye out for conflict exam form on Ed

# Regular Expressions

# Regular Expressions

Basis:

- $\varepsilon$ : The empty string itself matches the pattern (and nothing else does).
- $\varnothing$ : No strings match this pattern
- $a$ for any $a \in \Sigma$ : The character itself matching this pattern

Recursive:

- If $A, B$ are regular expressions then $(A \cup B)$ is a regular expression
  - matched by any string that matches $A$ or that matches $B$ [or both]
- If $A, B$ are regular expressions then $AB$ is a regular expression
  - matched by any string $x$ such that $x = yz$, $y$ matches $A$ and $z$ matches $B$
- If $A$ is a regular expression, then $A^*$ is a regular expression
  - matched by any string that can be divided into 0 or more strings that match $A$

# Regular Expressions

A regular expression is a recursively defined set of strings that form a language.

A regular expression will generate all strings in a language, and won't generate any strings that ARE NOT in the language

Hints:

- Come up with a few examples of strings that ARE and ARE NOT in your language
- Then, after you write your regex, check to make sure that it CAN generate all of your examples that are in the language, and it CAN'T generate those that are not

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least $k$ 1's.

Work on this problem with the people around you.

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**base-10 numbers:**

Our everyday numbers! Notice we have 10 symbols (0-9) to represent numbers.

256: $(\mathbf{2} * 10^2) + (\mathbf{5} * 10^1) + (\mathbf{6} * 10^0)$

**base-2 numbers:** (binary)

10: $(\mathbf{1} * 2^1) + (\mathbf{0} * 2^0)$

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

**All possible *strings* using numbers 0-9 that never start with 0**

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

**All possible *strings* using numbers 0-9 that never start with 0**

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

**All possible *strings* using numbers 0-9 that never start with 0**

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0” **is** a Base-10 number not considered

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

**All possible *strings* using numbers 0-9 that never start with 0**

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

**All possible *strings* using numbers 0-9 that never start with 0 *or* is 0**

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

**All possible *strings* using numbers 0-9 that never start with 0**

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

**All possible *strings* using numbers 0-9 that never start with 0 *or* is 0**

$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*)$

# Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

**Representing numbers all possible *strings* using numbers 0-9:**

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

**All possible *strings* using numbers 0-9 that never start with 0**

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

**All possible *strings* using numbers 0-9 that never start with 0 *or* is 0**

$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*)$

✅Generates only all possible Base-10 numbers

# Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

# Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

**Write a regular expression that matches all base-3 numbers**

# Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

**Write a regular expression that matches all base-3 numbers**

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

✅Generates only all possible Base-3 numbers

# Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

**Write a regular expression that matches all base-3 numbers**

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$$

Generates only all possible Base-3 numbers

**…divisible by 3**

# Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

**Write a regular expression that matches all base-3 numbers**

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

**…divisible by 3**

*Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)*

# Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

**Write a regular expression that matches all base-3 numbers**

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

**…divisible by 3**

*Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)*

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*\mathbf{0})$

✅all possible Base-3 numbers divisible by 3

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

$(0 \cup 1)^* \; 111 \; (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

$(0 \cup 1)^* \, 111 \, (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

**(0 ∪ 00 ∪ ε)** (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

**(0 ∪ 00 ∪ ε)** (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

**(0 ∪ 00 ∪ ε)** (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

**(0 ∪ 00 ∪ ε)** (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠ Generates “000” like “00 01 111”

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

**(0 ∪ 00 ∪ ε)** (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

✅ all binary strings with “111” and without “000”

# Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

**all binary strings that contain the substring “111”**

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

**…without the substring “000”**

Use careful case-work to modify this and produce only 0,1,or two 0’s

**(0 ∪ 00 ∪ ε)** (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ✅ all binary strings with “111” and without “000”

**(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)**

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

**Step 1:** Write out basic and more intricate cases

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

**Step 1:** Write out basic and more intricate cases

| Accepted Strings | Rejected Strings |
|---|---|
| ε | 00 |
| 1 | 11 |
| 10101 | 101011 |
| 0101 | 0100 |

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

**Step 1:** Write out basic and more intricate cases

| Accepted Strings | Rejected Strings |
|---|---|
| ε | 00 |
| 1 | 11 |
| 10101 | 101011 |
| 0101 | 0100 |

**Step 2**: Find a pattern!

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

**Step 1:** Write out basic and more intricate cases

| Accepted Strings | Rejected Strings |
|---|---|
| ε | 00 |
| 1 | 11 |
| 10101 | 10101**1** |
| 0101 | 010**0** |

**Step 2**: Find a pattern!

strings can be generated from **either a series** of **"01" or "10" substrings**

(1) Using the "01" substring, one additional 0 can be added

(1) Using the "10" substring, one additional 1 can be added

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

**Step 3:** Write out the expression with the two cases we found

# Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

**Step 3:** Write out the expression with the two cases we found

$((01)* (0 \cup \varepsilon)) \cup ((10)* (1 \cup \varepsilon))$

# Problem 1 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form $1^k y$ where $k \geq 1$ and $y \in \{0, 1\}^*$ has at least $k$ 1's.

# Problem 1 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form $1^k y$ where $k \geq 1$ and $y \in \{0, 1\}^*$ has at least $k$ 1's.

$1(0 \cup 1)^* 1(0 \cup 1)^*$

Explanation: While it may seem like we need to keep track of how many 1's there are, it turns out that we don't. Convince yourself that strings in the language are exactly those of the form 1x, where x is any binary string with at least one 1. Hence, x is matched by the regular expression $(0 \cup 1)^* 1(0 \cup 1)^*$

# Context-Free Grammars

# Context-Free Grammars

A context free grammar (CFG) is a finite set of production rules over:

- An alphabet Σ of “terminal symbols”
- A finite set $V$ of “nonterminal symbols”
- A start symbol (one of the elements of $V$) usually denoted $S$

A production rule for a nonterminal $A \in V$ takes the form

- $A \rightarrow w1 \mid w2 \mid \ldots \mid wk$

Where each $wi \in V \cup \Sigma^*$ is a string of nonterminals and terminals.

# Problem 2 – CFGs

Write a context-free grammar to match each of these languages.

a) All binary strings that start with 11.

b) All binary strings that contain at most one 1.

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Work on this problem with the people around you.

# Problem 2 – CFGs

a) All binary strings that start with 11.

# Problem 2 – CFGs

a) All binary strings that start with 11.

**Thinking back to regular expressions…**

# Problem 2 – CFGs

a) All binary strings that start with 11.

**Thinking back to regular expressions…**

11 (0 ∪ 1)*

# Problem 2 – CFGs

a) All binary strings that start with 11.

**Thinking back to regular expressions…**

11 (0 ∪ 1)*

**Now generate the CFG…**

# Problem 2 – CFGs

a) All binary strings that start with 11.

**Thinking back to regular expressions…**

11 (0 ∪ 1)*

**Now generate the CFG…**

**S** → 11**T**
**T** → 1**T** | 0**T** | **ε**

# Problem 2 – CFGs

b) All binary strings that contain at most one 1.

# Problem 2 – CFGs

b) All binary strings that contain at most one 1.

**Thinking back to Regular expressions…**

# Problem 2 – CFGs

b) All binary strings that contain at most one 1.

**Thinking back to Regular expressions…**

$0^* (1 \cup \varepsilon) 0^*$

# Problem 2 – CFGs

b) All binary strings that contain at most one 1.

**Thinking back to Regular expressions…**

$0^* (1 \cup \varepsilon) 0^*$

**Now generate the CFG…**

# Problem 2 – CFGs

b) All binary strings that contain at most one 1.

**Thinking back to Regular expressions…**

$0^* (1 \cup \varepsilon) 0^*$

**Now generate the CFG…**

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

# Problem 2 – CFGs

b) All binary strings that contain at most one 1.

**Thinking back to Regular expressions…**

$0^* (1 \cup \varepsilon) 0^*$

**Now generate the CFG…**

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

**Alternative solution:**

$S \rightarrow 0S \mid S0 \mid 1 \mid 0 \mid \varepsilon$

# Problem 2 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

# Problem 2 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

**S** → 01**S** | 10**S** | 0**S**1 | 1**S**0 | **S**01 | **S**10 | 2

# Problem 2 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

~~**S** → 01**S** | 10**S** | 0**S**1 | 1**S**0 | **S**01 | **S**10 | 2~~

Counter example: 00112

# Problem 2 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

~~**S** → 01**S** | 10**S** | 0**S**1 | 1**S**0 | **S**01 | **S**10 | 2~~

Counter example: 00112

Instead:

**S** → 2 | 2**T** | **T**2 | **ST** | **TS** | 0**S**1 | 1**S**0

**T** → **TT** | 0**T**1 | 1**T**0 | **ε**

# Deterministic Finite Automata

# Deterministic Finite Automata

- A DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.

- In other words:
  - Our machine is going to get a string as input. It will read one character at a time and update “its state.”
  - At every step, the machine thinks of itself as in one of the (finite number) vertices. When it reads the character, it follows the arrow labeled with that character to its next state.
  - Start at the “start state” (unlabeled, incoming arrow).
  - After you’ve read the last character, accept the string if and only if you’re in a “final state” (double circle).

- Every machine is defined with respect to an alphabet Σ
- Every state has exactly one outgoing edge for every character in Σ
- There is exactly one start state; can have as many accept states (aka final states) as you want – including none.

# Problem 3 – DFAs, Stage 1

Construct DFAs to recognize each of the following languages.
Let $\Sigma = \{0, 1, 2, 3\}$.

b) All strings whose digits sum to an even number.

c) All strings whose digits sum to an odd number.

# Problem 4 – DFAs, Stage 2

Construct DFAs to recognize each of the following languages.
Let $\Sigma = \{0, 1\}$.

c) All strings containing an even number of 1's and an odd number of 0's and not containing the substring 10.

Work on this problem with the people around you.

# Problem 3 – DFAs, Stage 1

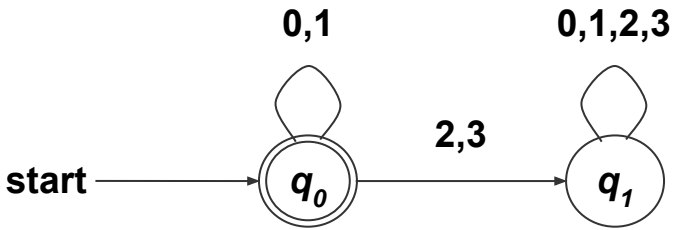
Let $\Sigma = \{0, 1, 2, 3\}$.

a) All binary strings.

# Problem 3 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

a) All binary strings.




$q_0$ : binary strings

$q_1$ : strings that contain a character which is not 0 or 1

# Problem 3 – DFAs, Stage 1

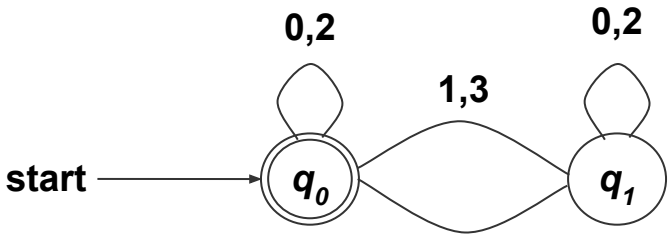
Let $\Sigma = \{0, 1, 2, 3\}$.

b) All strings whose digits sum to an even number.

# Problem 3 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.




$q_0$: strings whose sum of digits is even
$q_1$: strings whose sum of digits is odd

# Problem 3 – DFAs, Stage 1

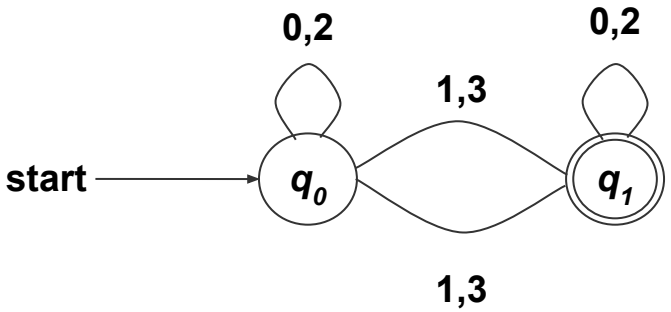
Let $\Sigma = \{0, 1, 2, 3\}$.

c) All strings whose digits sum to an odd number.

# Problem 3 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

c) All strings whose digits sum to an odd number.




$q_0$: strings whose sum of digits is even
$q_1$: strings whose sum of digits is odd

# Problem 4 – DFAs, Stage 2

Let $\Sigma = \{0, 1\}$.

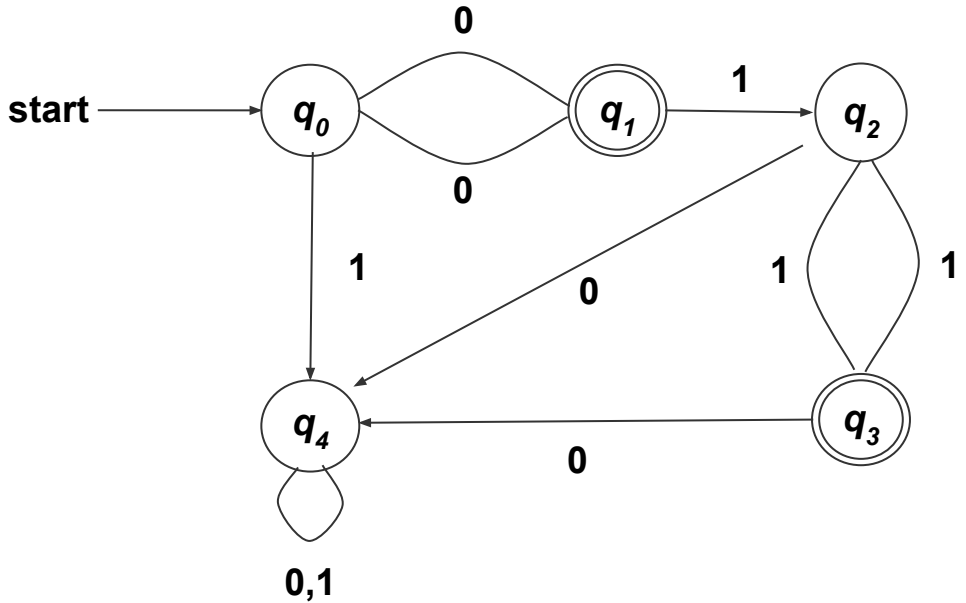
c) All strings containing an even number of 1's and an odd number of 0's and not containing the substring 10.

Work on this problem with the people around you.

# Problem 4 – DFAs, Stage 2

Let Σ = {0, 1}.

c) All strings containing an even number of 1's and an odd number of 0's and not containing the substring 10.




$q_0$: even 1s, even 0s (initial)
$q_1$: even 1s, odd 0s (accept!)
$q_2$: odd 1s, odd 0s
$q_3$: even 1s, odd 0s (accept!)
$q_4$: trap state - substring 10 occurred or we have an even number of 0's and a 1 already occurred

# All The Models

# Problem 5 – All The Models

Construct a valid regular expression, CFG, and DFA for the following languages.

a) All strings whose base-6 representation is divisible by 3 (leading zeros are ok). Let $\Sigma = \{0, 1, 2, 3, 4, 5\}$.

a) All binary strings of 0s capped by a 1 on either side

Work on this problem with the people around you.

# Problem 5 – All The Models

a) All strings whose base-6 representation is divisible by 3 (leading zeros are ok). Let $\Sigma = \{0, 1, 2, 3, 4, 5\}$.

# Problem 5 – All The Models

a) All strings whose base-6 representation is divisible by 3 (leading zeros are ok). Let Σ = {0, 1, 2, 3, 4, 5}.

Regular Expression: $(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5)^*(0 \cup 3)$

# Problem 5 – All The Models

a) All strings whose base-6 representation is divisible by 3 (leading zeros are ok). Let Σ = {0, 1, 2, 3, 4, 5}.

Regular Expression: $(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5)^*(0 \cup 3)$

CFG:

$$\mathbf{S} \rightarrow \mathbf{T}0 \mid \mathbf{T}3$$

$$\mathbf{T} \rightarrow 0\mathbf{T} \mid 1\mathbf{T} \mid 2\mathbf{T} \mid 3\mathbf{T} \mid 4\mathbf{T} \mid 5\mathbf{T} \mid \varepsilon$$

# Problem 5 – All The Models

a) All strings whose base-6 representation is divisible by 3 (leading zeros are ok). Let Σ = {0, 1, 2, 3, 4, 5}.

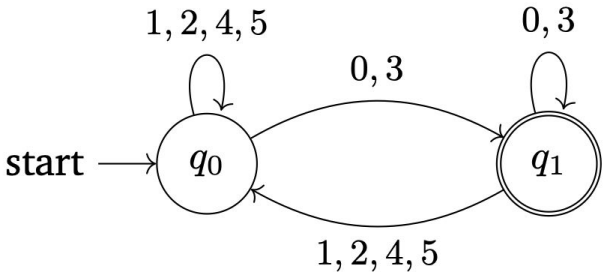
Regular Expression: $(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5)^*(0 \cup 3)$

CFG:

$$\mathbf{S} \rightarrow \mathbf{T}0 \mid \mathbf{T}3$$

$$\mathbf{T} \rightarrow 0\mathbf{T} \mid 1\mathbf{T} \mid 2\mathbf{T} \mid 3\mathbf{T} \mid 4\mathbf{T} \mid 5\mathbf{T} \mid \varepsilon$$

DFA:



# Problem 5 – All The Models

b) All binary strings of 0s capped by a 1 on either side

# Problem 5 – All The Models

b) All binary strings of 0s capped by a 1 on either side

Binary strings, therefore $\Sigma = \{0, 1\}$.

# Problem 5 – All The Models

b) All binary strings of 0s capped by a 1 on either side

Binary strings, therefore Σ = {0, 1}.

Regular Expression: $1(0)^*1$

# Problem 5 – All The Models

b) All binary strings of 0s capped by a 1 on either side

Binary strings, therefore Σ = {0, 1}.

Regular Expression: $1(0)^*1$

Context-Free Grammar:

$$\mathbf{S} \rightarrow 1\mathbf{T}1$$
$$\mathbf{T} \rightarrow 0\mathbf{T} \mid \varepsilon$$

# Problem 5 – All The Models

b) All binary strings of 0s capped by a 1 on either side

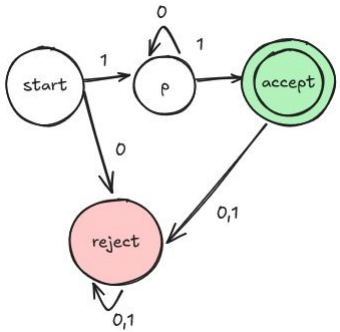
Binary strings, therefore Σ = {0, 1}.

Regular Expression: $1(0)^*1$

Context-Free Grammar:

$$\mathbf{S} \rightarrow 1\mathbf{T}1$$
$$\mathbf{T} \rightarrow 0\mathbf{T} \mid \varepsilon$$

Deterministic Finite Automata:



# Problem 5 – All The Models

b) All binary strings of 0s capped by a 1 on either side

Binary strings, therefore Σ = {0, 1}.

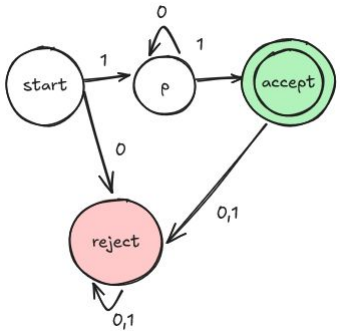
Regular Expression:

$$1(0)^*1$$

Context-Free Grammar:

$$\mathbf{S} \rightarrow 1\mathbf{T}1$$
$$\mathbf{T} \rightarrow 0\mathbf{T} \mid \varepsilon$$

Deterministic Finite Automata:




*What if we made the problem more complex?*

All binary strings of 0s capped by a **111000111** on either side

# Problem 5 – All The Models

b) All binary strings of 0s capped by a **111000111** on either side

Binary strings, therefore Σ = {0, 1}.

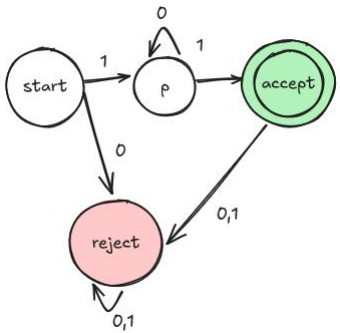
Regular Expression:

$$1(0)^*1$$

Context-Free Grammar:

$$\mathbf{S} \rightarrow 1\mathbf{T}1$$
$$\mathbf{T} \rightarrow 0\mathbf{T} \mid \varepsilon$$

Deterministic Finite Automata:



# Problem 5 – All The Models

b) All binary strings of 0s capped by a **111000111** on either side

Binary strings, therefore Σ = {0, 1}.

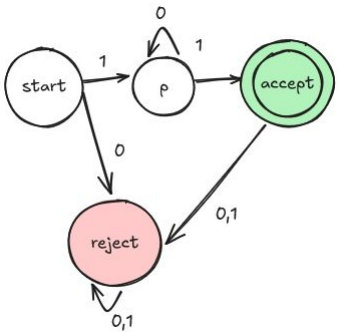
Regular Expression:

$$\textbf{111000111}\,(0)^*\,\textbf{111000111}$$

Context-Free Grammar:

$$\mathbf{S} \xrightarrow{111000111} \mathbf{T} \xrightarrow{111000111}$$
$$\mathbf{T} \rightarrow 0\mathbf{T} \mid \varepsilon$$

Deterministic Finite Automata:

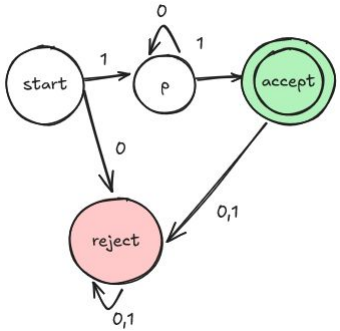



*Just like RE and CFG, what structure can we replace with complexity?*

# Problem 5 – All The Models

b) All binary strings of 0s capped by a **111000111** on either side

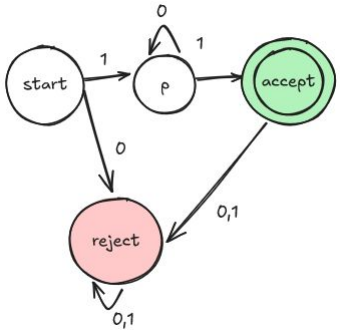
Deterministic Finite Automata:

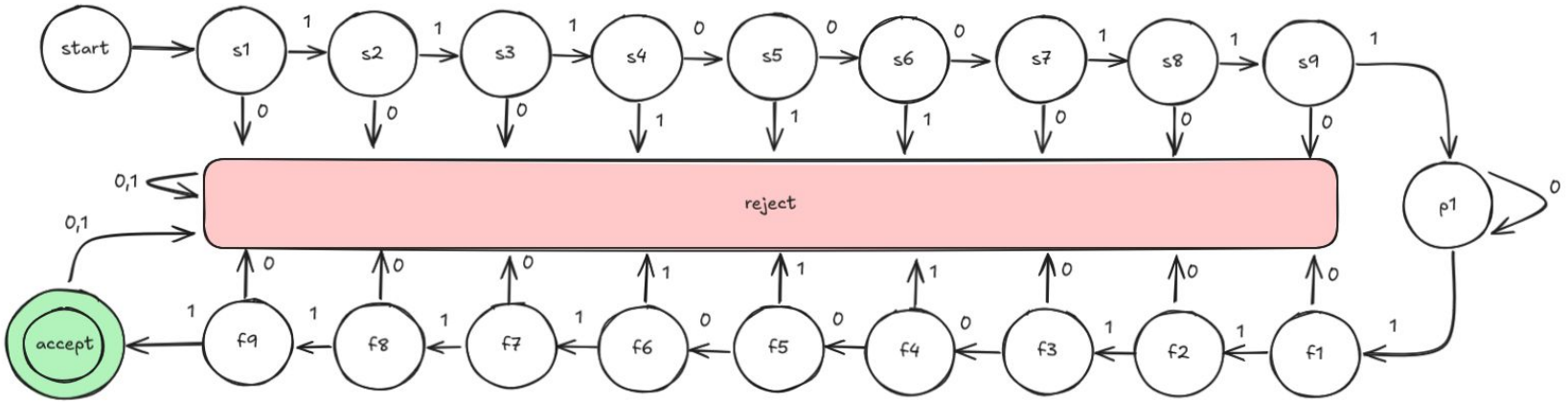


# Problem 5 – All The Models

b) All binary strings of 0s capped by a **111000111** on either side

Deterministic Finite Automata:



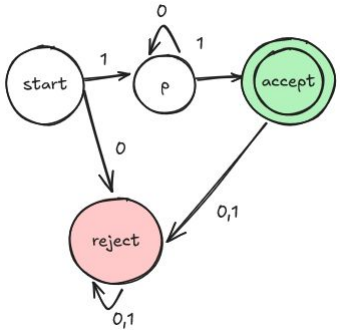
# Problem 5 – All The Models

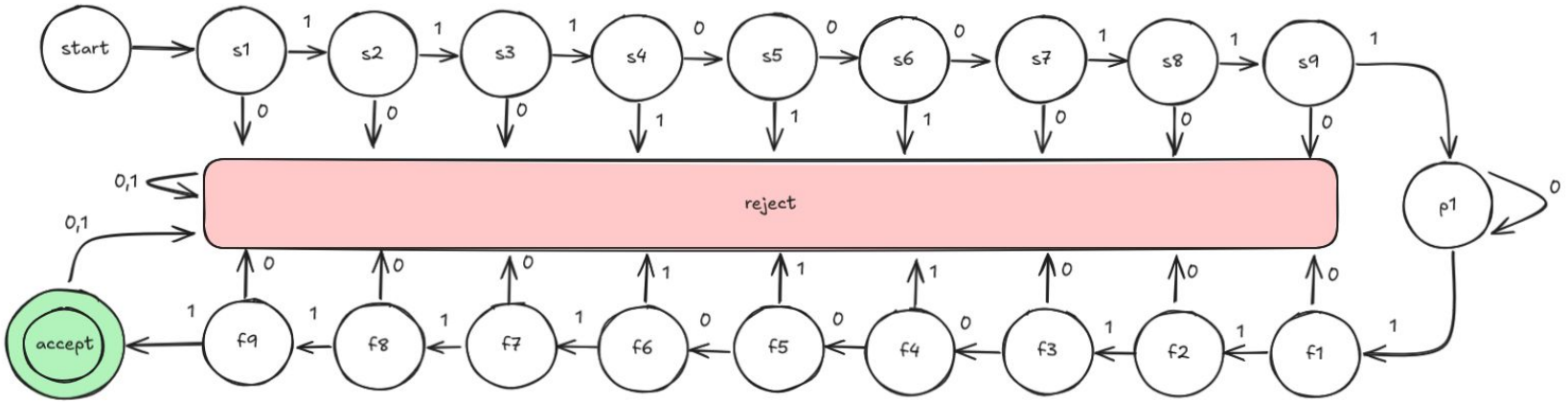
b) All binary strings of 0s capped by a **111000111** on either side

Deterministic Finite Automata:

*What structure can we replace with complexity?*

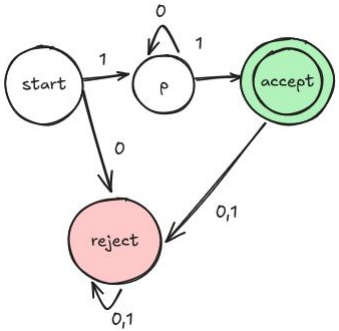
At most n 0s?



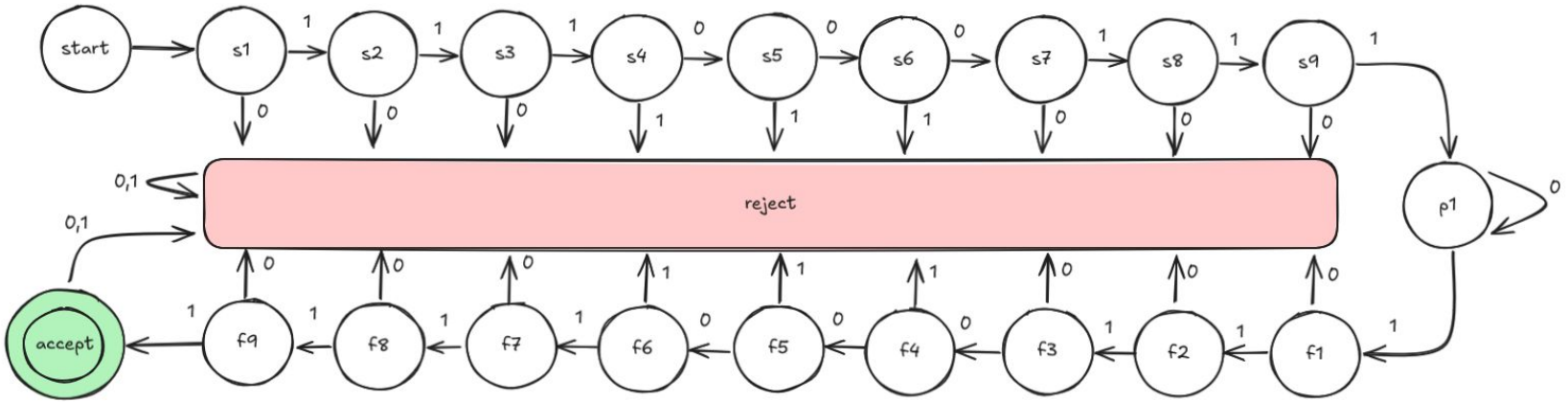



# Problem 5 – All The Models

b) All binary strings of 0s capped by a **111000111** on either side




**Takeaway:** start with top level structure, fill in finegrained details (like circuits)



# That's All, Folks!

**Thanks for coming to section this week!**
**Any questions?**

# Nondeterministic Finite Automata

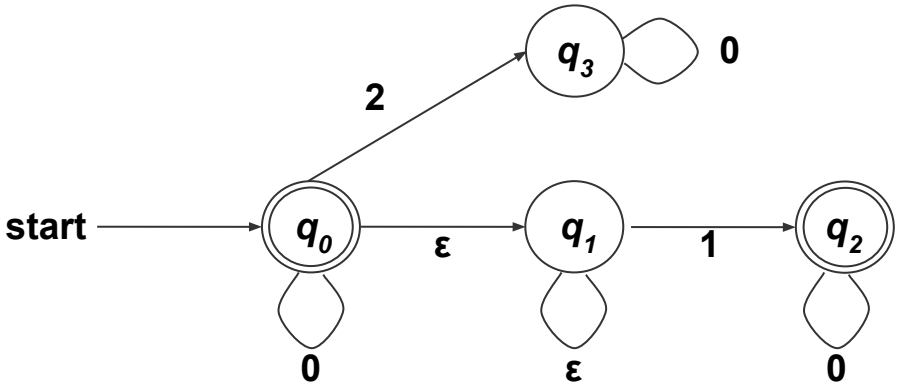
# Nondeterministic Finite Automata

- Similar to DFAs, but with less restrictions.
  - From a given state, we'll allow any number of outgoing edges labeled with a given character. (In a DFA, we have only 1 outgoing edge labeled with each character).
  - The machine can follow any of them.
  - We'll have edges labeled with "$\varepsilon$" – the machine (optionally) can follow one of those without reading another character from the input.
  - If we "get stuck" i.e. the next character is $a$ and there's no transition leaving our state labeled $a$, the computation dies.

- An NFA still has exactly one start state and any number of final states.
- The NFA accepts $x$ if there is some path from a start state to a final state labeled with $x$.
- From a state, you can have 0,1, or many outgoing arrows labeled with a single character. You can choose any of them to build the required path.

# Problem 6 – NFAs

a) What language does the following NFA accept?

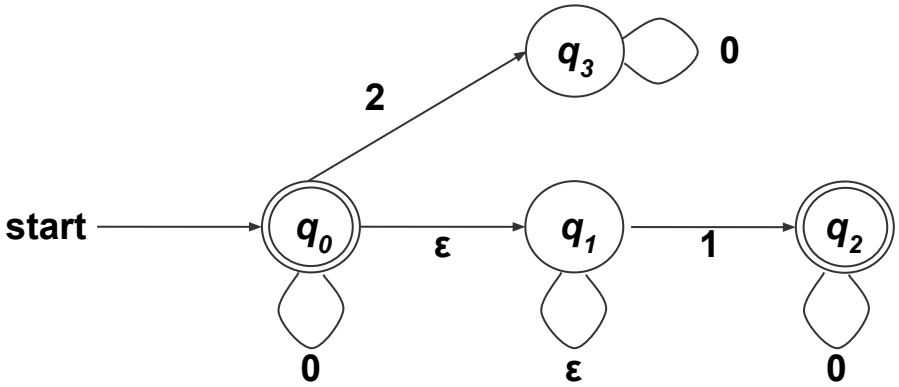



b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Work on this problem with the people around you.

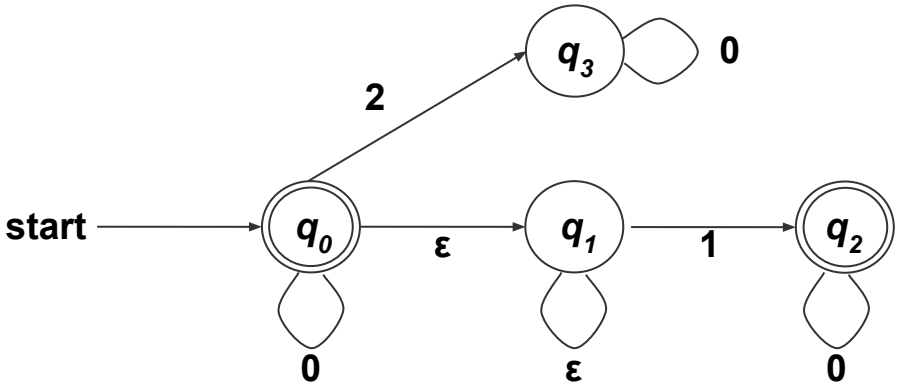
# Problem 6 – NFAs

a) What language does the following NFA accept?



# Problem 6 – NFAs

a) What language does the following NFA accept?




All strings of only 0's and 1's, not containing more than one 1.

# Problem 6 – NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

# Problem 6 – NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.