Week 8: Regular Languages and Finite Automaton

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcement

- Midterm 2: 6:00~8:00pm EX 100 (July 17)
  - Bonus Question (midterm 2). Course material (up to and including W7)
  - Missed Exam? Submit request by July 22 (documents by July 25)
- Structure for finite automaton section:
  - Week 8: DFA, NFA, and regular expressions
  - Week 9: Proof of correctness of finite automatons, equivalence of DFA-NFA-regular expressions
  - Week 10: Limitations of regular languages, pumping lemma
- Tutorials this week: more examples of DFAs and NFAs

Regular Languages

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

	English	**Boolean**	**Hexadecimal**
Alphabet			

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

A **string** w over Σ is a finite sequence of symbols from Σ. The empty string is denoted ϵ.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$
String			

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

A **string** w over Σ is a finite sequence of symbols from Σ. The empty string is denoted ϵ.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$
String	*BANANA*	01011001	*FFFF*00

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

A **string** w over Σ is a finite sequence of symbols from Σ. The empty string is denoted ϵ.

The **length** of w is the number of characters it contains.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$
String (Length)	*BANANA*	01011001	*FFFF*00

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

A **string** w over Σ is a finite sequence of symbols from Σ. The empty string is denoted ϵ.

The **length** of w is the number of characters it contains.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$
String (Length)	*BANANA (6)*	01011001 *(8)*	*FFFF*00 *(6)*

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

A **string** w over Σ is a finite sequence of symbols from Σ. The empty string is denoted ϵ.

The **length** of w is the number of characters it contains.

A **language** L over Σ is a subset of Σ^*. $\emptyset$ is the empty language.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$
String (Length)	*BANANA (6)*	01011001 *(8)*	*FFFF*00 *(6)*
Language			

General Definitions of Languages

An **alphabet** Σ is a finite set of symbols.

A **string** w over Σ is a finite sequence of symbols from Σ. The empty string is denoted ϵ.

The **length** of w is the number of characters it contains.

A **language** L over Σ is a subset of Σ^*. $\emptyset$ is the empty language.

	English	**Boolean**	**Hexadecimal**
Alphabet	$\{A, .., Z\}$	$\{0, 1\}$	$\{0, ..., F\}$
String (Length)	*BANANA (6)*	01011001 *(8)*	*FFFF*00 *(6)*
Language	all words	$\{0, 1\}^*$	$\{c \in \Sigma^* : \lvert c\rvert = 6\}$

Operations on Languages

- **Union**: Let I and J be languages over the alphabet Σ, then the language
$$I \cup J = \{x \in \Sigma^* : x \in I \text{ or } x \in J\}.$$
- **Concatenation**: Let I and J be as the above, then the language
$$I \circ J = \{xy \in \Sigma^* : x \in I, y \in J\}$$

Operations on Languages

- **Union**: Let I and J be languages over the alphabet Σ, then the language
$$I \cup J = \{x \in \Sigma^* : x \in I \text{ or } x \in J\}.$$
- **Concatenation**: Let I and J be as the above, then the language
$$I \circ J = \{xy \in \Sigma^* : x \in I, y \in J\}$$
- **Exponentiation**: Let I be as above, then for any $k \in \mathbb{N}$
$$I^k = \underbrace{I \circ \cdots \circ I}_{k}$$

Operations on Languages

- **Union**: Let I and J be languages over the alphabet Σ, then the language
$$I \cup J = \{x \in \Sigma^* : x \in I \text{ or } x \in J\}.$$
- **Concatenation**: Let I and J be as the above, then the language
$$I \circ J = \{xy \in \Sigma^* : x \in I, y \in J\}$$
- **Exponentiation**: Let I be as above, then for any $k \in \mathbb{N}$
$$I^k = \underbrace{I \circ \cdots \circ I}_{k}$$
Question: what is L^0?

Operations on Languages

- **Union**: Let I and J be languages over the alphabet Σ, then the language
$$I \cup J = \{x \in \Sigma^* : x \in I \text{ or } x \in J\}.$$
- **Concatenation**: Let I and J be as the above, then the language
$$I \circ J = \{xy \in \Sigma^* : x \in I, y \in J\}$$
- **Exponentiation**: Let I be as above, then for any $k \in \mathbb{N}$
$$I^k = \underbrace{I \circ \cdots \circ I}_{k}$$
Question: what is L^0? Answer: $\{\epsilon\}$.

Operations on Languages

- **Union**: Let I and J be languages over the alphabet Σ, then the language
$$I \cup J = \{x \in \Sigma^* : x \in I \text{ or } x \in J\}.$$
- **Concatenation**: Let I and J be as the above, then the language
$$I \circ J = \{xy \in \Sigma^* : x \in I, y \in J\}$$
- **Exponentiation**: Let I be as above, then for any $k \in \mathbb{N}$
$$I^k = \underbrace{I \circ \cdots \circ I}_{k}$$
Question: what is L^0? Answer: $\{\epsilon\}$.
- **Kleene Star**: Let I be as above, then
$$I^{\circledast} = \{x \in \Sigma^* : x \in I^k \text{ for any } k \in \mathbb{N}\}$$

$I = \{w \in \{0,1\}^* : w = \underbrace{0 \cdots 0}_{n}, n \in \mathbb{N}\}$

$J = \{w \in \{0,1\}^* : w = \underbrace{1 \cdots 1}_{n}, n \in \mathbb{N}\}$

$I \cup J = \{w \in \{0,1\}^* : w = \underbrace{0 \cdots 0}_{n}$ OR $w = \underbrace{1 \cdots 1}_{n}, n \in \mathbb{N}\}$

Regular Languages

Regular Language

A *regular language* over an alphabet Σ is defined recursively as:

Regular Languages

Regular Language

A *regular language* over an alphabet Σ is defined recursively as:

- $\emptyset$, the empty set, is a regular language.
- $\{\epsilon\}$, the language consisting of an empty string, is a regular language.
- For any character, a.k.a *symbol*, $a \in \Sigma$, $\{a\}$ is a regular language.

Regular Languages

Regular Language

A *regular language* over an alphabet Σ is defined recursively as:

- $\emptyset$, the empty set, is a regular language.
- $\{\epsilon\}$, the language consisting of an empty string, is a regular language.
- For any character, a.k.a *symbol*, $a \in \Sigma$, $\{a\}$ is a regular language.
- (*Closure of Regular Languages.*) If I and J are regular languages, then so are $I \cup J$, $I \circ J$, and $I^{\circledast}$.

Regular Languages

Regular Language

A *regular language* over an alphabet Σ is defined recursively as:

- $\emptyset$, the empty set, is a regular language.
- $\{\epsilon\}$, the language consisting of an empty string, is a regular language.
- For any character, a.k.a *symbol*, $a \in \Sigma$, $\{a\}$ is a regular language.
- (*Closure of Regular Languages.*) If I and J are regular languages, then so are $I \cup J$, $I \circ J$, and $I^{\circledast}$.

Example.

1. Σ^* (take the Kleene Star of $\{a \in \Sigma\}$).
2. Over the Boolean alphabet $\Sigma = \{0, 1\}$, the language

$$L = \{w \in \Sigma : w = \epsilon \text{ or } w = 01 \cdots 01\}.$$

Regular Expressions

Overview

Consider the following language L:

$$L = \{w \in \{0,1\}^* : w \text{ starts with 0 or 1, followed by any number of ones, and ends with a 0}\}$$

Overview

Consider the following language L:

$$L = \{w \in \{0,1\}^* : w \text{ starts with 0 or 1,} \\ \text{followed by any number of ones,} \\ \text{and ends with a 0}\}$$

What we want is a *template* of the form:

(0 or 1) and then (any number of 1) and then 0

Overview

Consider the following language L:

$$L = \{w \in \{0,1\}^* : w \text{ starts with 0 or 1,} \\ \text{followed by any number of ones,} \\ \text{and ends with a 0}\}$$

What we want is a *template* of the form:

(0 or 1) and then (any number of 1) and then 0

... or even more succinctly still

$$(0+1)1^*0$$

Regular Expressions

Let R and S be regular expressions. Then valid operations include:

$R + S$: formally known as **alternation**, but can be read as *or*. Analogous to the $\cup$ operation on languages.

RS: formally known as **concatenation**. Analogous to the $\circ$ operation.

R^*: formally known as **repetition**. Analogous to the $\circledast$ operation.

Regular Expressions

Let R and S be regular expressions. Then valid operations include:

$R + S$: formally known as **alternation**, but can be read as *or*. Analogous to the $\cup$ operation on languages.

RS: formally known as **concatenation**. Analogous to the $\circ$ operation.

R^*: formally known as **repetition**. Analogous to the $\circledast$ operation.

Regular Expression

A *regular expression* over an alphabet Σ is defined recursively. $\emptyset$, ϵ, and $a \in \Sigma$ are regular expressions. If R and S are regular expression over Σ, then so are $(R + S)$, (RS), and R^*.

Regular Expressions

Let R and S be regular expressions. Then valid operations include:

$R + S$: formally known as **alternation**, but can be read as *or*. Analogous to the $\cup$ operation on languages.

RS: formally known as **concatenation**. Analogous to the $\circ$ operation.

R^*: formally known as **repetition**. Analogous to the $\circledast$ operation.

Regular Expression

A *regular expression* over an alphabet Σ is defined recursively. $\emptyset$, ϵ, and $a \in \Sigma$ are regular expressions. If R and S are regular expression over Σ, then so are $(R + S)$, (RS), and R^*.

Regular expression R **represents** the language $\mathcal{L}(R)$ (in the previous example $L = \mathcal{L}((0 + 1)1^*0)$).

Finite Automaton

Deterministic Finite Automaton (DFA)

Definition

A *Deterministic Finite (State) Automaton* (DFA) M is the quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.

Deterministic Finite Automaton (DFA)

Definition

A *Deterministic Finite (State) Automaton* (DFA) M is the quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.
- Σ is a finite **alphabet**.

Deterministic Finite Automaton (DFA)

Definition

A *Deterministic Finite (State) Automaton* (DFA) M is the quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.
- Σ is a finite **alphabet**.
- $\delta : Q \times \Sigma \to Q$ is the **transition function**. In essence δ encodes the transition from state to state. Given the current state q and an input symbol a, $\delta(q, a) = p$ is the state where you end up.

Deterministic Finite Automaton (DFA)

Definition

A *Deterministic Finite (State) Automaton* (DFA) M is the quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.
- Σ is a finite **alphabet**.
- $\delta : Q \times \Sigma \to Q$ is the **transition function**. In essence δ encodes the transition from state to state. Given the current state q and an input symbol a, $\delta(q, a) = p$ is the state where you end up.
- s is *the* start or **initial** state.

Deterministic Finite Automaton (DFA)

Definition

A *Deterministic Finite (State) Automaton* (DFA) M is the quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.
- Σ is a finite **alphabet**.
- $\delta : Q \times \Sigma \to Q$ is the **transition function**. In essence δ encodes the transition from state to state. Given the current state q and an input symbol a, $\delta(q, a) = p$ is the state where you end up.
- s is *the* start or **initial** state.
- $F \subset Q$ is the *set* of **accepting states**.

Deterministic Finite Automaton (DFA)

Definition

A *Deterministic Finite (State) Automaton* (DFA) M is the quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.
- Σ is a finite **alphabet**.
- $\delta : Q \times \Sigma \to Q$ is the **transition function**. In essence δ encodes the transition from state to state. Given the current state q and an input symbol a, $\delta(q, a) = p$ is the state where you end up.
- s is *the* start or **initial** state.
- $F \subset Q$ is the *set* of **accepting states**.

DFA M **accepts** the language $\mathcal{L}(M)$.

DFA Example

Consider the following DFA $M = (Q, \{0, 1\}, \delta, s, F)$ where:

- $Q = \{q_0, q_1, q_2, q_3\}$.
- $s = q_0$.
- $F = \{q_3\}$.

State	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_3

Table 1: Transition table for δ.

DFA Example

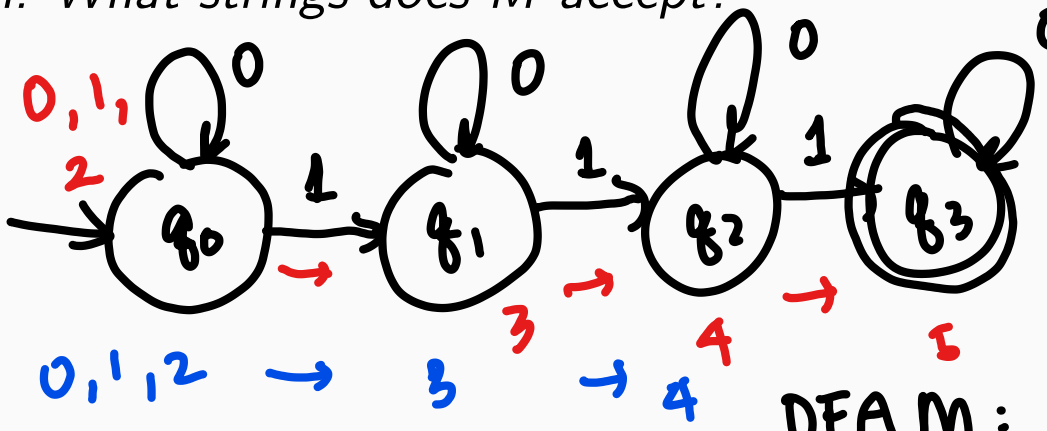
Consider the following DFA $M = (Q, \{0, 1\}, \delta, s, F)$ where:

- $Q = \{q_0, q_1, q_2, q_3\}$.
- $s = q_0$.
- $F = \{q_3\}$.

State	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_3

Table 1: Transition table for δ.

Question: What strings does M accept?



S = 00111 accepted by M

S_2 = 0011 NOT accepted by M

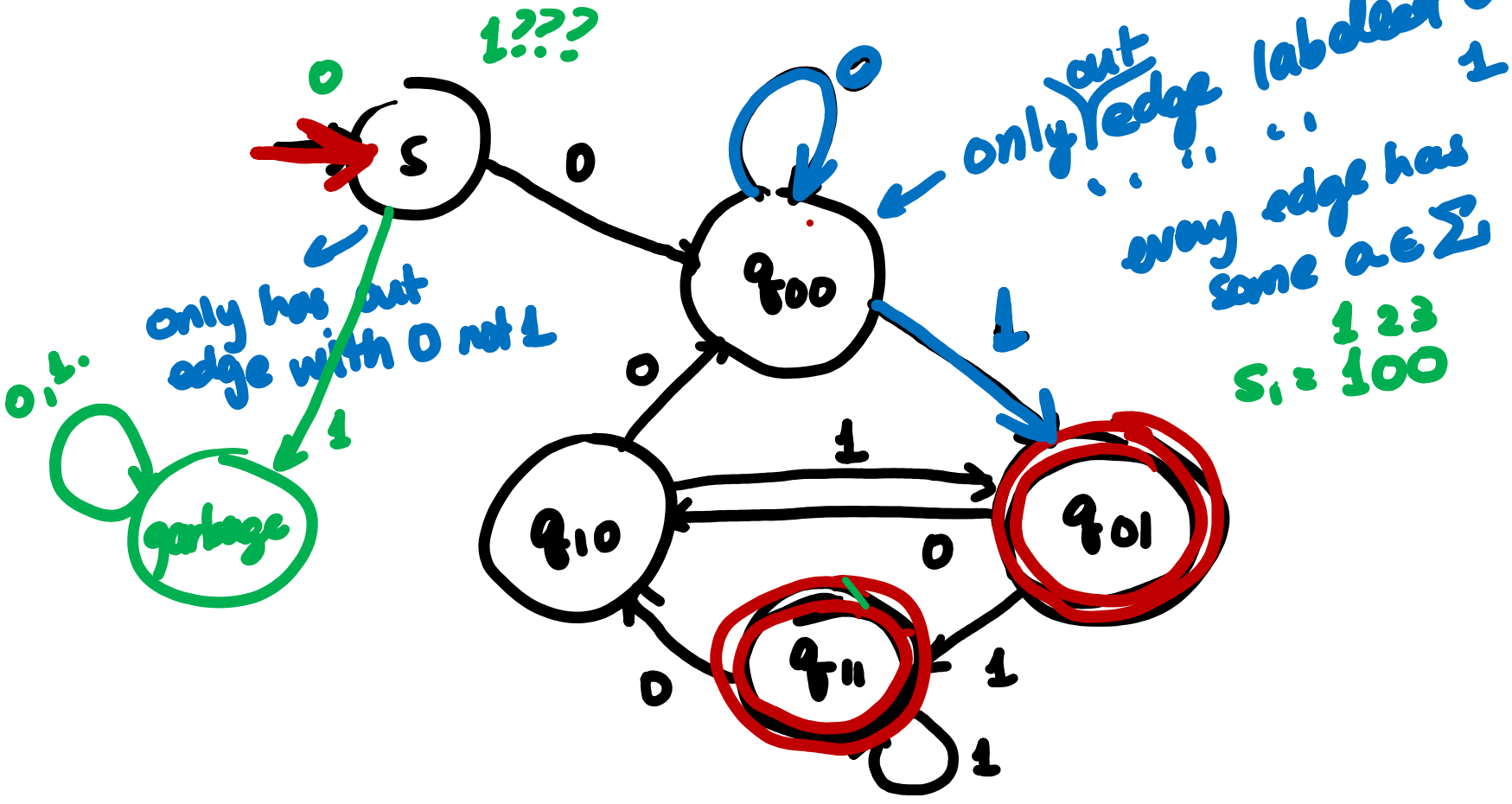
DFA M: accepts strings with ≥ 3 "1"s.

Now you try!

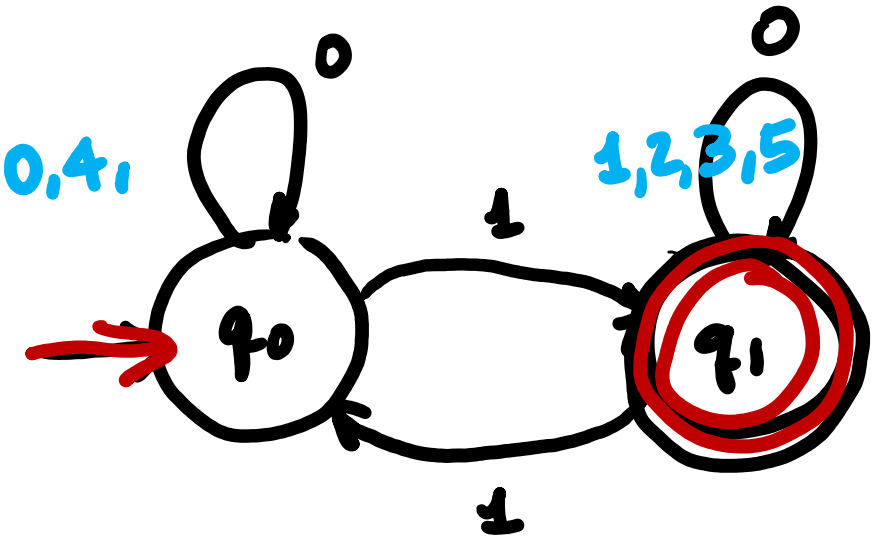
Design a DFA which accepts the following languages

1. $L_1 = \{w \in \{0,1\}^* : w \text{ starts with } 0 \text{ and ends with } 1\}$.
2. $L_2 = \{w \in \{0,1\}^* : w \text{ contains an odd number of } 1\}$.
3. $L_3 = \{0^n 1^m : m, n \in \mathbb{N}, m + n \text{ is even}\}$.

$L_1 = \{w \in \{0,1\}^* : w \text{ starts with } 0 \text{ and ends with } 1\}.$

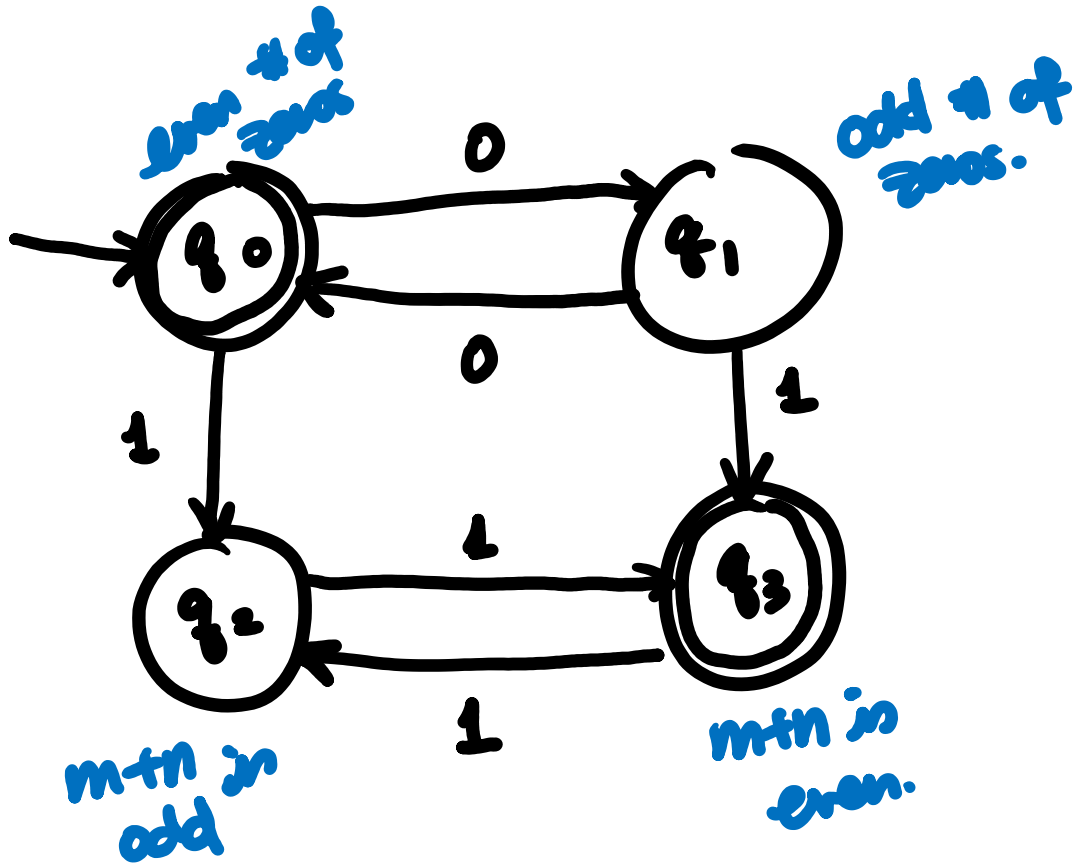


$L_2 = \{w \in \{0,1\}^* : w \text{ contains an odd number of 1}\}.$



1 2 3 4 5
$S_1 =$ 1 0 0 1 1

$L_3 = \{0^n 1^m : m, n \in \mathbb{N}, m + n \text{ is even}\}.$



Intuition

First some intuition for what an automaton is: consider *Manufactoria*.

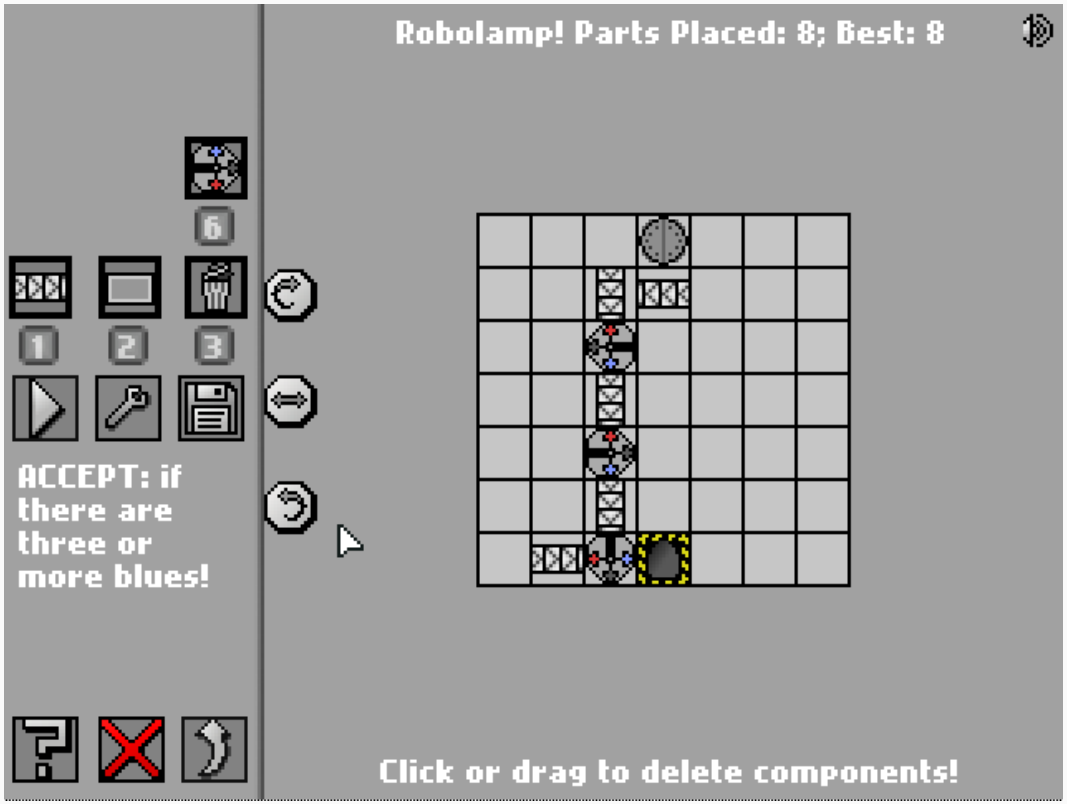


Figure 1: Putting robots in their place!

Non-Deterministic Finite Automaton (NFA)

A Non-Deterministic Finite Automaton (NDFA or NFA) is similar to a DFA, but allows for *non-determinism*.

Non-Deterministic Finite Automaton (NFA)

A Non-Deterministic Finite Automaton (NDFA or NFA) is similar to a DFA, but allows for *non-determinism*.

NFA

An NFA is a quintuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q is a *finite set* of **states**.
- Σ is a finite **alphabet**.
- $\delta : Q \times \{\Sigma \cup \{\epsilon\}\} \to \mathcal{P}(Q)$ is the **transition function** where $\mathcal{P}(Q)$ is the *power set* of Q (the set of all subsets of Q). That is to say: δ outputs a set of states instead of a single state.
- s is *the* start or **initial** state.
- $F \subset Q$ is the *set* of **accepting states**.

NFA Example

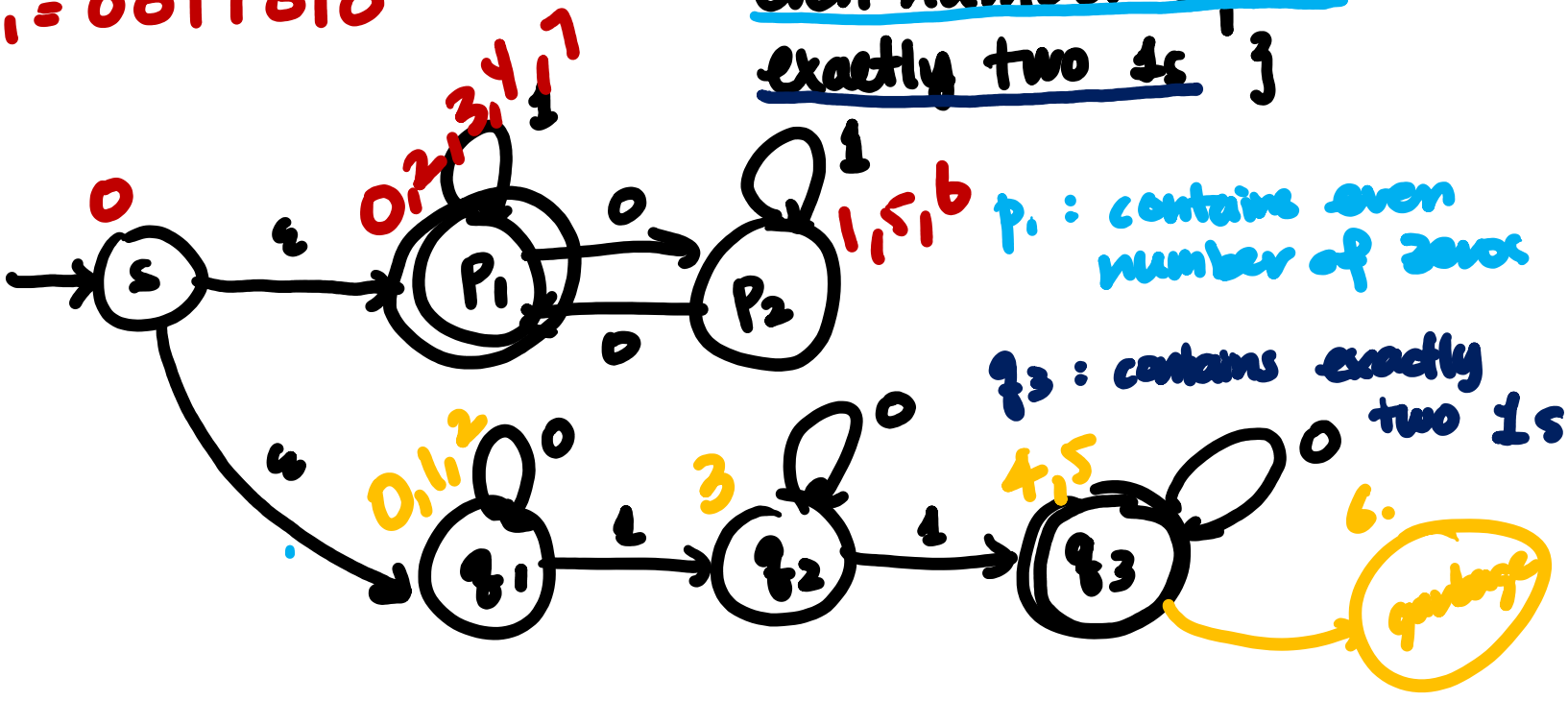
We want to construct an NDFA M which accepts language:

$L = \{w \in \{0,1\}^* : w \text{ contains an even number of 0s or exactly two 1s}\}$.

Construct NFA M which accepts:

$L = \{ w \in \{0,1\}^* : w$ contains an even number of 0s or exactly two 1s $\}$

$s_1 = 0\,0\,1\,1\,0\,1\,0$ (positions 1 2 3 4 5 6 7)



RECAP:

1. Language L : over alphabet Σ (Σ^* : all strings over alphabet)
   - strings : composed of $a \in \Sigma$
   - L is <u>set of strings</u> ($L \subseteq \Sigma^*$)
   - OPS: Union ($\cup$), concat ($\circ$), Kleene Star ($*$)

2. Regular Expressions — alternation ($+$), concat ($\cdot$), repetition ($*$) reps.

3. FINITE AUTOMATONS : DFA, NFA

DFA	NFA
each state $q \in Q$ has 1 transition for each $a \in \Sigma$	$q \in Q$ has any # of transitions on $a \in \Sigma$
ε-transition NOT allowed	ε-transition allowed