SE 3310b

Theoretical Foundations of Software Engineering

Week 1:

Introduction and Determisitic Finite Automata

Aleksander Essex

Western Engineering

Introduction

Course Introduction

- Course website: `http://essex.cc/3310`
- Review of lecture schedule, marking scheme, late policy
- Course textbook: Michael Sipser. Introduction to the Theory of Computation, 3rd Edition, Cengage Learning, 2012.

What is this course about?

Why learn about the **theory of computation**? As an aspiring software engineer you can imagine it may be useful to have some understanding of the fundamental capabilities and limitations of computers and software. In this course we will explore some of the most fundamental questions of our field:

- What is computation?
- What do you need to perform a computation?
- Which problems can you solve with computation?
- Of the problems with solutions, which are easy, and which are hard?

Background and Significance

From Cogs to Q-bits

The Early Days:

- 1822: Charles Babbage dreams of a general purpose machine to perform calculations. Inspired by the idea, his colleague Ada Lovelace outlines the first software program.
- 1900-28: Mathematician David Hilbert proposes a set of 23 the greatest (as-of-then) unsolved problems in mathematics. The 2nd and 10th of Hilbert's problems eventually lead to him proposing the Entscheidungsproblem (the *decision problem*). Loosely speaking, the question asks whether there is an algorithmic means of *deciding* whether a proof exists for a given logical statement, or not.
- 1936: Alan Turing and Alonzo Church independently prove "*no*" to the Entscheidungsproblem problem, introducing a universal model of computation in so doing.

From Cogs to Q-bits

Recent Times:

- 1956: Noam Chomsky describes a hierarchy of formal grammars with important implications for programming languages.
- 1971: University of Toronto professor Stephen Cook proposes the P vs. NP, considered one of the most important open problems in computer science today.
- 1994: Peter Shor proposes to run on a quantum computer that would efficiently solve the integer factorization problem, which would break RSA and related public-key cryptosystems currently used to secure digital communications worldwide—if you could build such a device. Recent documents show that, among others, NSA is working hard to develop one.

Notation

General notation

- The integers: $\mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$
- The natural numbers: $\mathbb{N} = \{1, 2, \ldots\}$
- Set-builder notation: $\{x \mid statement\}$. Translation: the set of all x's for which $statement$ is true.
  - Some use ":" instead of "|"
  - What is $\{x \in \mathbb{Z} : x > 0\}$?

Sets

A set is an unordered collection of elements.
Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

Sets

A set is an unordered collection of elements.
Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

- Element Order: Does $\{x_1, x_2, x_3, x_4\} = \{x_4, x_3, x_2, x_1\}$?

Sets

A set is an unordered collection of elements.

Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

- Element Order: Does $\{x_1, x_2, x_3, x_4\} = \{x_4, x_3, x_2, x_1\}$?
- Set membership: Which statement is true?
  - $x_1 \in B$
  - $x_1 \notin B$

Sets

A set is an unordered collection of elements.

Let $A = \{x_1, x_2, x_3, x_4\}$, $B = \{x_4, x_5, x_6\}$

- Element Order: Does $\{x_1, x_2, x_3, x_4\} = \{x_4, x_3, x_2, x_1\}$?
- Set membership: Which statement is true?
  - $x_1 \in B$
  - $x_1 \notin B$
- Subsets: Which statements are true?
  - $x_1 \subset A$
  - $x_1 \subset A$
  - $x_1 \subseteq A$
  - $\{x_1, x_2, x_3, x_4\} \subset A$
  - $\{x_1, x_2, x_3, x_4\} \subseteq A$

Sets

A set is an unordered collection unique of elements.

Let $A = \{x_1, x_2, x_3, x_4\}$

- Power Set: Set of all subsets.
  - What is $\mathcal{P}(A)$?

Sets

A set is an unordered collection unique of elements.

Let $A = \{x_1, x_2, x_3, x_4\}$

- Power Set: Set of all subsets.
  - What is $\mathcal{P}(A)$?
- $|\mathcal{P}| = 2^{|A|}$

Set Operations

Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

- Union of sets $A \cup B = \{x \,|\, x \in A \text{ or } x \in B\}$
  - What is $A \cup B$?

Set Operations

Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

- Union of sets $A \cup B = \{x \,|\, x \in A \text{ or } x \in B\}$
  - What is $A \cup B$?
- Intersection of sets: $A \cap B = \{x \,|\, x \in A \text{ and } x \in B\}$
  - What is $A \cap B$?

Set Operations

Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

- Union of sets $A \cup B = \{x \,|\, x \in A \text{ or } x \in B\}$
  - What is $A \cup B$?
- Intersection of sets: $A \cap B = \{x \,|\, x \in A \text{ and } x \in B\}$
  - What is $A \cap B$?
- Cardinality of a set: $|A|$ the number of elements in A
  - What is $|A|$?
  - Is $|B| > |A|$?
  - Let $A = \{\{1, 2\}\}$. What is $|A|$?

Set Operations

Let $A = \{x_1, x_2, x_3, x_4\}, B = \{x_4, x_5, x_6\}$

- Union of sets $A \cup B = \{x \,|\, x \in A \text{ or } x \in B\}$
  - What is $A \cup B$?
- Intersection of sets: $A \cap B = \{x \,|\, x \in A \text{ and } x \in B\}$
  - What is $A \cap B$?
- Cardinality of a set: $|A|$ the number of elements in A
  - What is $|A|$?
  - Is $|B| > |A|$?
  - Let $A = \{\{1, 2\}\}$. What is $|A|$?
    - A is a set that contains *one* element, the set $\{1, 2\}$.
  - Let $C = \{\varnothing\}$. What is $|C|$?

Set Operations

Let $A = \{x_1, x_2, x_3, x_4\}$, $B = \{x_4, x_5, x_6\}$

- Union of sets $A \cup B = \{x \,|\, x \in A \text{ or } x \in B\}$
  - What is $A \cup B$?
- Intersection of sets: $A \cap B = \{x \,|\, x \in A \text{ and } x \in B\}$
  - What is $A \cap B$?
- Cardinality of a set: $|A|$ the number of elements in A
  - What is $|A|$?
  - Is $|B| > |A|$?
  - Let $A = \{\{1, 2\}\}$. What is $|A|$?
    - A is a set that contains *one* element, the set $\{1, 2\}$.
  - Let $C = \{\varnothing\}$. What is $|C|$?
    - First observe $|\varnothing| = 0$, i.e., the empty set is a set containing *no* elements. On the other hand $|C| = |\{\varnothing\}| = 1$, because C is a set containing one element; the empty set.

Tuples

A tuple is an ordered list of (not necessarily unique) elements.
Let $A = (x_1, x_2, x_3, x_4), B = (x_4, x_5, x_6)$

Tuples

A tuple is an ordered list of (not necessarily unique) elements.
Let $A = (x_1, x_2, x_3, x_4)$, $B = (x_4, x_5, x_6)$

- Element Order: Does $(x_1, x_2, x_3, x_4) = (x_4, x_3, x_2, x_1)$?

Tuples

A tuple is an ordered list of (not necessarily unique) elements.
Let $A = (x_1, x_2, x_3, x_4), B = (x_4, x_5, x_6)$

- Element Order: Does $(x_1, x_2, x_3, x_4) = (x_4, x_3, x_2, x_1)$?
- Cartesian product: $A \times B = \{(a, b) : a \in A, b \in B\}$
  - What is $A \times B$?

Tuples

A tuple is an ordered list of (not necessarily unique) elements.
Let $A = (x_1, x_2, x_3, x_4), B = (x_4, x_5, x_6)$

- Element Order: Does $(x_1, x_2, x_3, x_4) = (x_4, x_3, x_2, x_1)$?
- Cartesian product: $A \times B = \{(a, b) : a \in A, b \in B\}$
  - What is $A \times B$?
- Cartesian product of multiple sets:
  $A \times B \times \ldots \times Z = \{(a, b, \ldots z) : a \in A, b \in B \ldots z \in Z\}$

Tuples

A tuple is an ordered list of (not necessarily unique) elements.
Let $A = (x_1, x_2, x_3, x_4), B = (x_4, x_5, x_6)$

- Element Order: Does $(x_1, x_2, x_3, x_4) = (x_4, x_3, x_2, x_1)$?
- Cartesian product: $A \times B = \{(a, b) : a \in A, b \in B\}$
  - What is $A \times B$?
- Cartesian product of multiple sets:
  $A \times B \times \ldots \times Z = \{(a, b, \ldots z) : a \in A, b \in B \ldots z \in Z\}$
- $|A \times B| = |A| \cdot |B|$
  - What is $|A \times B|$?

Functions

A function, denoted:

$$f\colon \mathbb{A} \rightarrow \mathbb{B}$$

is a mapping between a set of inputs $\mathbb{A}$ and a set of outputs $\mathbb{B}$.

Functions

A function, denoted:

$$f\colon \mathbb{A} \to \mathbb{B}$$

is a mapping between a set of inputs $\mathbb{A}$ and a set of outputs $\mathbb{B}$.

- A function's domain is the set of possible inputs, $\mathbb{A}$.

Functions

A function, denoted:

$$f\colon \mathbb{A} \to \mathbb{B}$$

is a mapping between a set of inputs $\mathbb{A}$ and a set of outputs $\mathbb{B}$.

- A function's domain is the set of possible inputs, $\mathbb{A}$.
- A function's range is the set of outputs, and can be used to describe one of two related terms:
  - The codomain is a set $\mathbb{B}$ in to which all the outputs fall
  - The image: the subset $\mathbb{B}' \subset \mathbb{B}$ of the codomain into which all the elements of the domain are mapped.

Functions

A function, denoted:

$$f\colon \mathbb{A} \to \mathbb{B}$$

is a mapping between a set of inputs $\mathbb{A}$ and a set of outputs $\mathbb{B}$.

- A function's domain is the set of possible inputs, $\mathbb{A}$.
- A function's range is the set of outputs, and can be used to describe one of two related terms:
  - The codomain is a set $\mathbb{B}$ in to which all the outputs fall
  - The image: the subset $\mathbb{B}' \subset \mathbb{B}$ of the codomain into which all the elements of the domain are mapped.
- Example: Let $f\colon \mathbb{Z} \to \mathbb{Z}$ be $f\colon x \mapsto x^2$, i.e., $f(x) = x^2$.
  - The codomain is $\mathbb{Z}$
  - The image is $\mathbb{N}^0$ (i.e., $\{0\} \cup \mathbb{N}$)

Functions

A function, denoted:

$$f\colon \mathbb{A} \to \mathbb{B}$$

is a mapping between a set of inputs $\mathbb{A}$ and a set of outputs $\mathbb{B}$.

- A function's domain is the set of possible inputs, $\mathbb{A}$.
- A function's range is the set of outputs, and can be used to describe one of two related terms:
  - The codomain is a set $\mathbb{B}$ in to which all the outputs fall
  - The image: the subset $\mathbb{B}' \subset \mathbb{B}$ of the codomain into which all the elements of the domain are mapped.
- Example: Let $f\colon \mathbb{Z} \to \mathbb{Z}$ be $f\colon x \mapsto x^2$, i.e., $f(x) = x^2$.
  - The codomain is $\mathbb{Z}$
  - The image is $\mathbb{N}^0$ (i.e., $\{0\} \cup \mathbb{N}$)
- Multiple variables. Example: $f\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ where $f\colon x, y \mapsto x^2 + y^2$, *i.e.*, $f(x, y) = x^2 + y^2$

Alphabets and Strings

An alphabet Σ is a set of symbols.

- $\Sigma = \{0, 1\}$ is the binary alphabet
- Hey, what's this alphabet: $\Sigma = \{A, B, C \ldots X, Y, Z\}$?

Alphabets and Strings

An alphabet Σ is a set of symbols.

- $\Sigma = \{0, 1\}$ is the binary alphabet
- Hey, what's this alphabet: $\Sigma = \{A, B, C \ldots X, Y, Z\}$?

A string is a finite sequence of symbols of a given alphabet.

- Let $\Sigma = \{A, B, C \ldots Z\}$
  - Is "PASSWORD" a valid string?
  - Is "PASSWORD1" a valid string?

Alphabets and Strings

An alphabet Σ is a set of symbols.

- $\Sigma = \{0, 1\}$ is the binary alphabet
- Hey, what's this alphabet: $\Sigma = \{A, B, C \ldots X, Y, Z\}$?

A string is a finite sequence of symbols of a given alphabet.

- Let $\Sigma = \{A, B, C \ldots Z\}$
  - Is "PASSWORD" a valid string?
  - Is "PASSWORD1" a valid string?
- Let $\Sigma = \{0, 1\}$
  - Is "0" a valid string?
  - Is "$0000000000000000000000000000$" a valid string?
  - Is $\underbrace{000 \ldots 000}_{\text{Infinity times}}$ a valid string?

Alphabets and Strings

An alphabet Σ is a set of symbols.

- $\Sigma = \{0, 1\}$ is the binary alphabet
- Hey, what's this alphabet: $\Sigma = \{A, B, C \ldots X, Y, Z\}$?

A string is a finite sequence of symbols of a given alphabet.

- Let $\Sigma = \{A, B, C \ldots Z\}$
  - Is "PASSWORD" a valid string?
  - Is "PASSWORD1" a valid string?
- Let $\Sigma = \{0, 1\}$
  - Is "0" a valid string?
  - Is "$0000000000000000000000000000$" a valid string?
  - Is $\underbrace{000 \ldots 000}_{\text{Infinity times}}$ a valid string?

Alphabets and Strings

The symbol ε denotes the empty string. It functions as an identity element (think of adding 0, or multiplying by 1)

- What is $|\varepsilon|$?
- What is $0\varepsilon 1$?
- What is $\varepsilon 0\varepsilon\varepsilon\varepsilon 1$?

Alphabets and Strings

Powers of an alphabet: Σ^k is the set of strings of length k.

Alphabets and Strings

Powers of an alphabet: Σ^k is the set of strings of length k.

- Let $\Sigma = \{0, 1\}$
  - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^1 = \{0, 1\}$
  - $\Sigma^2 = \{\ldots\}$?

Alphabets and Strings

Powers of an alphabet: Σ^k is the set of strings of length k.

- Let $\Sigma = \{0, 1\}$
  - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^1 = \{0, 1\}$
  - $\Sigma^2 = \{\ldots\}$?
- Kleene Star: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$

Alphabets and Strings

Powers of an alphabet: Σ^k is the set of strings of length k.

- Let $\Sigma = \{0, 1\}$
  - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^1 = \{0, 1\}$
  - $\Sigma^2 = \{\ldots\}$?
- Kleene Star: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$
- For $\Sigma = \{0, 1\}$, what is Σ^*?

Alphabets and Strings

Powers of an alphabet: Σ^k is the set of strings of length k.

- Let $\Sigma = \{0, 1\}$
  - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^1 = \{0, 1\}$
  - $\Sigma^2 = \{\ldots\}$?
- Kleene Star: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$
- For $\Sigma = \{0, 1\}$, what is Σ^*?
- Fact: Although Σ^* is infinite, all elements of Σ^* have finite length.

Alphabets and Strings

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

- Let $a = 00$
- Let $b = 11$
- $ab = 0011$

Alphabets and Strings

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

- Let $a = 00$
- Let $b = 11$
- $ab = 0011$
- $(ab)^2 = abab = ?$

Alphabets and Strings

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

- Let $a = 00$
- Let $b = 11$
- $ab = 0011$
- $(ab)^2 = abab = ?$
- $a^2b^2 = aabb = ?$

Formal Language

A formal language over alphabet Σ, denoted L, is a subset $L \subseteq \Sigma^*$. It's basically a set of strings. The kinds of formal languages we're interested in are ones that can be produced by some kind of rule.

Formal Language

A formal language over alphabet Σ, denoted L, is a subset $L \subseteq \Sigma^*$. It's basically a set of strings. The kinds of formal languages we're interested in are ones that can be produced by some kind of rule. Let $\Sigma = \{0, 1\}$:

- What language is $\{0, 10, 00, 000, 010, 100, 110, \dots\}$?

Formal Language

A formal language over alphabet Σ, denoted L, is a subset $L \subseteq \Sigma^*$. It's basically a set of strings. The kinds of formal languages we're interested in are ones that can be produced by some kind of rule. Let $\Sigma = \{0, 1\}$:

- What language is $\{0, 10, 00, 000, 010, 100, 110, \dots\}$? It's the language of strings that end in 0.

Formal Language

A formal language over alphabet Σ, denoted L, is a subset $L \subseteq \Sigma^*$. It's basically a set of strings. The kinds of formal languages we're interested in are ones that can be produced by some kind of rule. Let $\Sigma = \{0, 1\}$:

- What language is $\{0, 10, 00, 000, 010, 100, 110, \dots\}$? It's the language of strings that end in 0.
- What language is $\{1, 01, 10, 11, 001, 010, 011, 100, \dots\}$?

Formal Language

A formal language over alphabet Σ, denoted L, is a subset $L \subseteq \Sigma^*$. It's basically a set of strings. The kinds of formal languages we're interested in are ones that can be produced by some kind of rule. Let $\Sigma = \{0, 1\}$:

- What language is $\{0, 10, 00, 000, 010, 100, 110, \dots\}$? It's the language of strings that end in 0.
- What language is $\{1, 01, 10, 11, 001, 010, 011, 100, \dots\}$? It's the language of strings that have at least one 1.

Formal Language

A formal language over alphabet Σ, denoted L, is a subset $L \subseteq \Sigma^*$. It's basically a set of strings. The kinds of formal languages we're interested in are ones that can be produced by some kind of rule. Let $\Sigma = \{0, 1\}$:

- What language is $\{0, 10, 00, 000, 010, 100, 110, \dots\}$? It's the language of strings that end in 0.
- What language is $\{1, 01, 10, 11, 001, 010, 011, 100, \dots\}$? It's the language of strings that have at least one 1.

How does this differ from a *natural* language, like English? How is it similar?

What is Computation?

You might intuitively expect computation to be posed in a kind of "ask a question, get an answer" fashion. For example, if you had an equation $y = x^2$ and I say "let $x = 3$, find y" then computation would mean putting 3 into a black box and getting 9 out.

That's how computation occurs *in effect*. But for this course, and indeed for the purposes of studying computation, we're going to pose questions of computation as *decision problems*. In our parabola example that's like asking: yes or no, is the point $(3, 9)$ on the curve?

What is Computation?

Informally: Computation is modeled as answers to *yes/no* questions.

What is Computation?

More formally, computation means:

- Given the description of some language L
- Given an input string s
- Compute a *yes* or *no* answer to the following question:
  - Is $s \in L$?
- That is, does L *accept* s, or it *does not* accept s.

Computation, therefore, is the act of *deciding* if $s \in L$, or not.

Deterministic Finite Automata (DFA)

Deterministic Finite Automata (DFA)

Definition 1 (Deterministic Finite Automaton (DFA))**.**

A *deterministic finite automaton* is a $5-$tuple $(Q, \Sigma, \delta, q_0, F)$

1. Q: a finite set of **states**
2. Σ: a finite **alphabet**
3. $\delta: Q \times \Sigma \rightarrow Q$: a **transition function**
4. $q_0 \in Q$: the **starting state**
5. $F \subseteq Q$: a set of **accepting states**

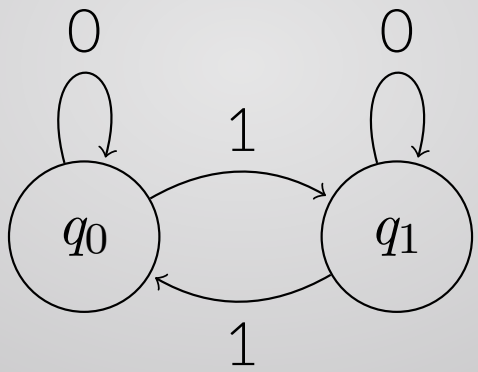
Deterministic Finite Automata (DFA): An Example

States (Q):

$$Q = \{q_0, q_1\}$$

Deterministic Finite Automata (DFA): An Example

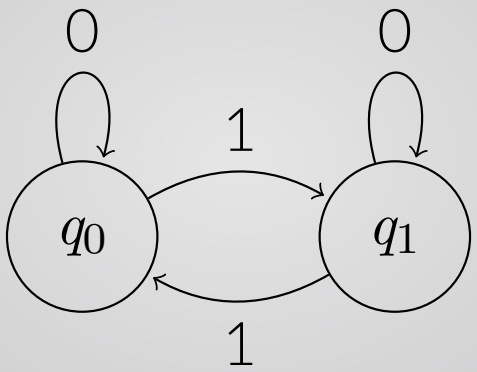
Alphabet (Σ):



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

Deterministic Finite Automata (DFA): An Example

Transition Function (δ):

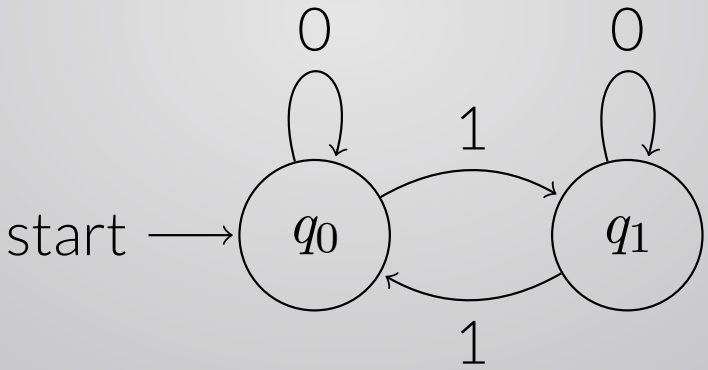


$\delta =$

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

Deterministic Finite Automata (DFA): An Example

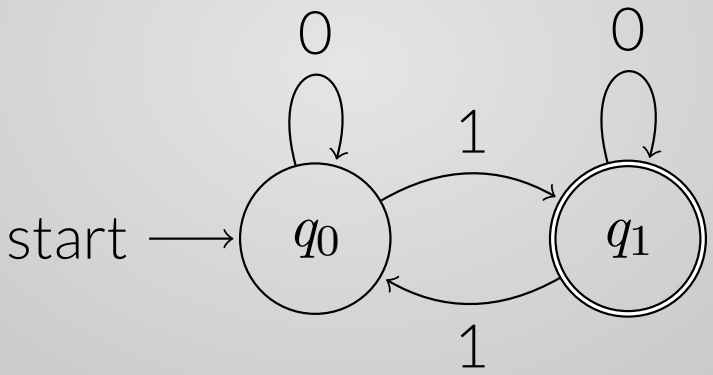
Initial state (q_0):



q_0 = (see above)

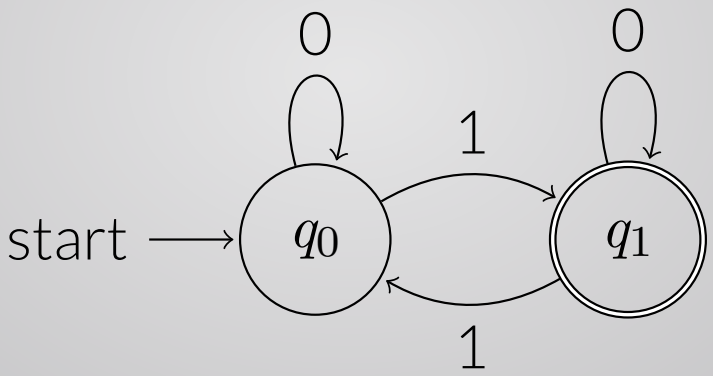
Deterministic Finite Automata (DFA): An Example

Accepting state(s) (F):



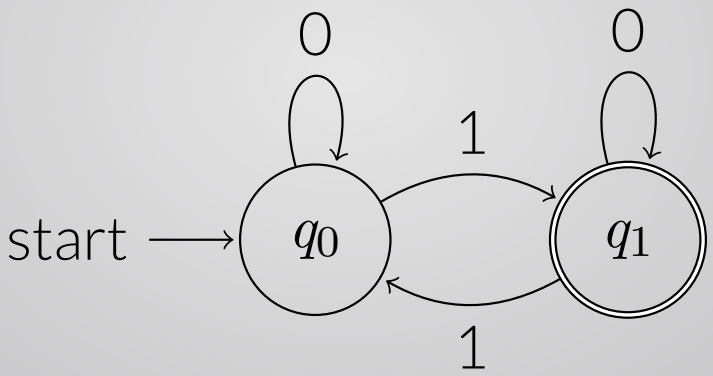
$$F = \{q_1\}$$

Deterministic Finite Automata (DFA): An Example



What *language* does this accept?

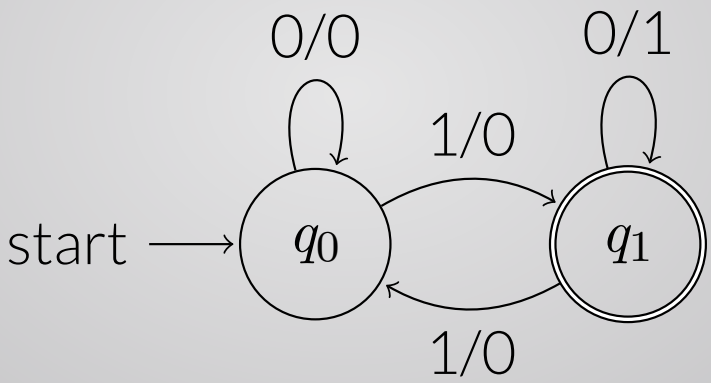
Deterministic Finite Automata (DFA): An Example



How is this model different from, say, a Mealy machine?

Finite State Transducer (FST)

A finite state transducer is a DFA for which output is defined.



A Mealy machine is an finite state transducer.

Finite State Transducer (FST)

- For now we will focus only on *yes/no* problems (hard enough as it is)
  - That is, an automaton either accepts a given string, or does not accept it.
- The notion of output will become clearer when we talk about Turing machines
  - Turing machines have a "tape" that can leave behind a result