CSC 320
Foundations of Computer Science

Lecture 2

Instructor: Dr. Ulrike Stege

Territory Acknowledgement

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

This meeting will be recorded

"Please be aware our sessions are being screen-recorded to allow students who are not able to attend to watch later and will be posted in Brightspace."

Deadlines; Assessment

10%	25%	25%	40%
Quizzes	Assignments	Midterms	Final Exam
Quiz 1-8: 1% each Quiz 9: 2%	Assignment 1-5: 5% each	Midterm 1: 10% Midterm 2: 15%	

May

S	M	T	W	T	F	S
			3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

June

S	M	T	W	T	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

July

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Timed quizzes (~30 min)
Review before starting quiz

Office hours

- Mondays, Thursdays 2:30-3:30

Last time ....

- countable vs uncountable

Countable & uncountable

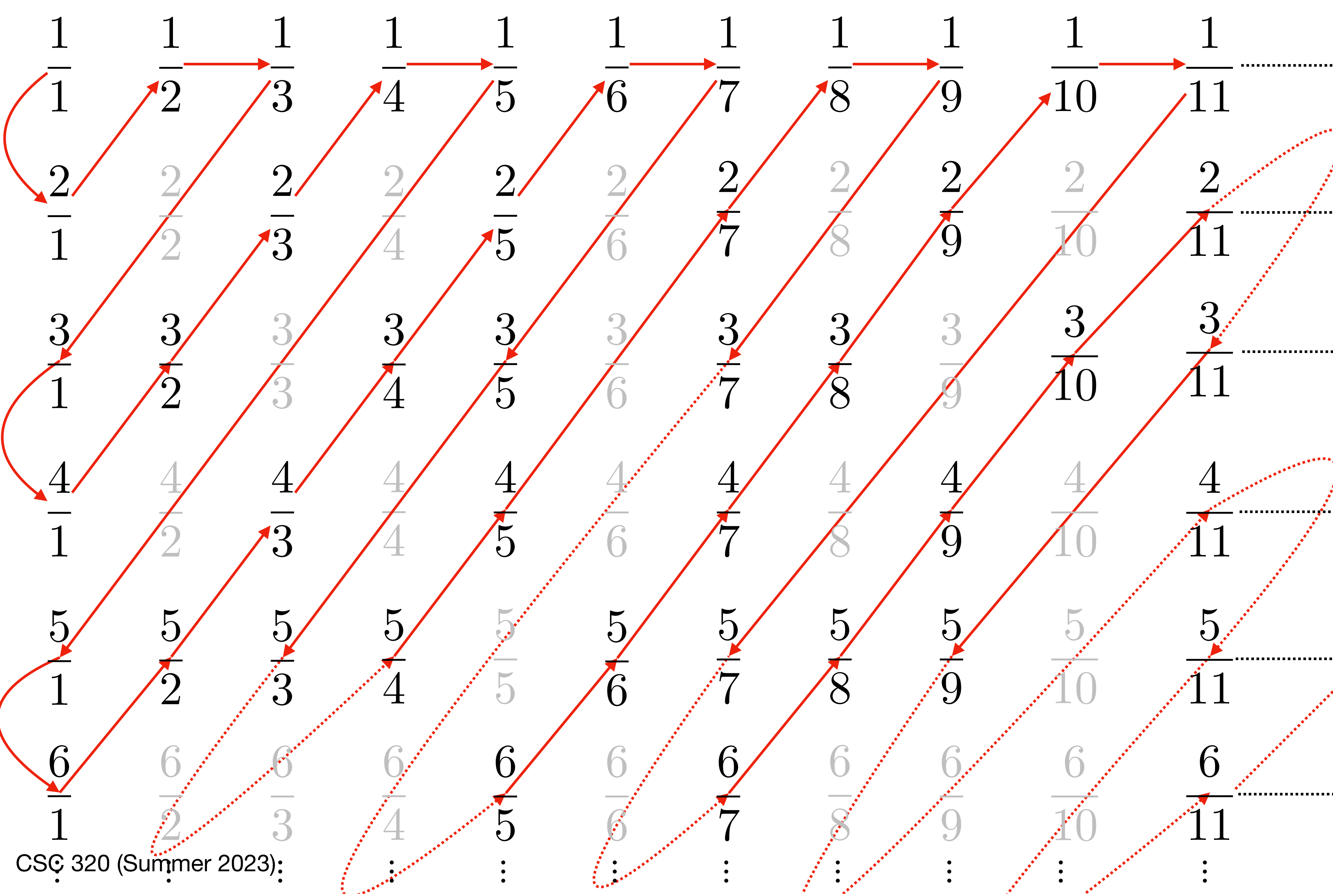
- A set is **countable** if it is **finite** or **countably infinite**: the elements of a **countable** set can always be counted one at a time; *every element of the set is associated with a unique natural number*

- A set that is neither finite nor countably infinite is **uncountable**

$\mathbb{Z}$ is countable

$\mathbb{N}$	$\mathbb{Z}$
0	0
1	-1
2	1
3	-2
4	2
5	-3
6	3
...	...

Counting the set of positive rational numbers $\mathbb{Q}^+ \setminus \{0\}$

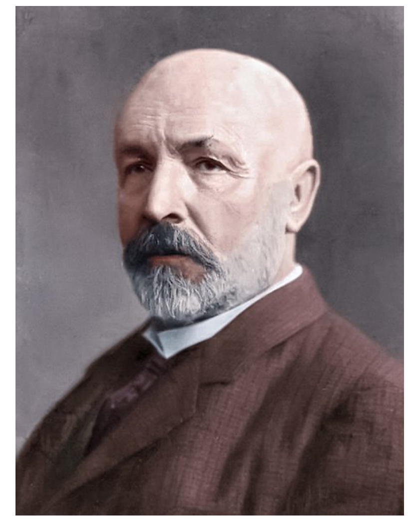


Theorem: $\mathbb{R}$ is uncountable
Proof uses Cantor's Diagonalization Argument

Proof by contradiction. Assume: $\mathbb{R}$ is countable.

- Then we can *enumerate* $\mathbb{R}$
- We can also enumerate each subset of $\mathbb{R}$
- Therefore, if $\mathbb{R}$ is countable then we can enumerate the set of all real numbers between 0 and 1:

$$
\begin{aligned}
x_1 &= 0.d_{11}d_{12}d_{13}d_{14}\ldots \\
x_2 &= 0.d_{21}d_{22}d_{23}d_{24}\ldots \\
x_3 &= 0.d_{31}d_{32}d_{33}d_{34}\ldots \\
x_4 &= 0.d_{41}d_{42}d_{43}d_{44}\ldots \\
&\vdots \qquad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots
\end{aligned}
$$



George Cantor
1845—1918

where $$x_n = 0.d_{n1}d_{n2}d_{n3}d_{n4}\ldots$$

Cantor's Diagonalization Argument (continued)

$$
\begin{array}{ll}
x_1 & 0.d_{11}d_{12}d_{13}d_{14}\ldots \\
x_2 & 0.d_{21}d_{22}d_{23}d_{24}\ldots \\
x_3 & 0.d_{31}d_{32}d_{33}d_{34}\ldots \\
x_4 & 0.d_{41}d_{42}d_{43}d_{44}\ldots \\
 & \;\vdots\quad\;\vdots\quad\;\vdots\quad\;\vdots\quad\;\vdots \\
x_n = & 0.d_{n1}d_{n2}d_{n3}d_{n4}\ldots \\
 & \;\vdots\quad\;\vdots\quad\;\vdots\quad\;\vdots\quad\;\vdots
\end{array}
$$

$c = x_1$?

No: $c_1 \neq d_{11}$

Consider the number $c = 0.c_1c_2c_3c_4\ldots$ with $c_i \neq d_{ii}$ for each i

Clearly: $0 \leq c < 1$

$\boxed{c \textit{ is not} \text{ in above list}}$

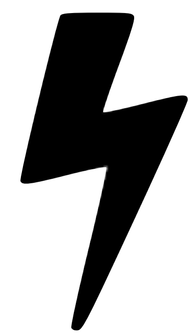
Cantor's Diagonalization Argument (continued)

$$
\begin{array}{llr}
x_1 & 0.d_{11}d_{12}d_{13}d_{14}\ldots & c \neq x_1 \\
x_2 & 0.d_{21}d_{22}d_{23}d_{24}\ldots & c \neq x_2 \\
x_3 & 0.d_{31}d_{32}d_{33}d_{34}\ldots & c \neq x_3 \\
x_4 & 0.d_{41}d_{42}d_{43}d_{44}\ldots & c \neq x_4 \\
 & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & \\
x_n = & 0.d_{n1}d_{n2}d_{n3}d_{n4}\ldots & c \neq x_n \\
 & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots &
\end{array}
$$

$$c = 0.c_1c_2c_3c_4\ldots \qquad c_i \neq d_{ii}$$

Cantor's Diagonalization Argument (continued)

- Answer: No!
  $0.c_1c_2c_3c_4\dots$ is not in the list as it is different from each list element! Therefore, since $0 \leq c < 1$, the list **does not** contain all real numbers between 0 and 1



- If we cannot even enumerate this subset of $\mathbb{R}$, then it is also impossible to enumerate $\mathbb{R}$

Recommended reading: https://www.cantorsparadise.com/hilberts-hotel-an-ingenious-explanation-of-infinity-1d1a79932080

Terminology Review: Sets

- **Set** $S = \{3, \ell, 20, green, \alpha\}$
  - Objects in a set: **elements/members**
- **Membership/Non-membership** $\alpha \in S; \beta \notin S$
- **Empty set** $\emptyset$
  - Set with zero members/elements
- **Singleton set**
  - Set with exactly one member
- **Unordered pair**
  - Set with exactly two members

Terminology Review: Set Operations

- **Union** of sets A and B: $A \cup B = \{x | x \in A \text{ or } x \in B\}$
- **Intersection** of sets A and B: $A \cap B = \{x | x \in A \text{ and } x \in B\}$
- **Complement** of set A: $\bar{A} = \{x | x \notin A\}$
- **Set difference** of sets A and B: $A - B = \{x \in A : x \notin B\}$

  $A \backslash B$

More terminology: Powerset

- **Powerset** $\mathscr{P}(A)$ of set A: set of all subsets of A
  $\mathscr{P}(A) = \{S \mid S \subseteq A\}$. Note that $\emptyset \in \mathscr{P}(A)$

Alphabets, Languages, Strings, Symbols

Terminology to describe and work with finite automata and more

- An **alphabet** Σ is a **finite** set of **symbols**
  - eg: binary alphabet $\{0,1\}$ or the Roman/Latin alphabet
- A **string** over an alphabet Σ is a finite sequence of symbols from Σ
  - 0001 is a string over alphabet $\{0,1\}$
- The **empty string** ε is the string with no symbols

Alphabets, Languages, Strings, Symbols (2)

- The **set of all strings** over an alphabet Σ is denoted Σ^*
- Note: $\epsilon \in \Sigma^*$
  - ie, ϵ is a string over any alphabet
- Example: Let $\Sigma = \{a,b\}$. Then $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, \ldots\}$

Alphabets, Languages, Strings, Symbols (3)

- The **length** $|w|$ of a string $w \in \Sigma^*$ is the number of symbols of w when considered as a sequence, e.g.
  - length of the empty string ϵ: $|\epsilon| = 0$
  - $w = ab$, $|w| = 2$
- For a string $w \in \Sigma^*$, the symbol in the i^{th} position of w is denoted w_i. We say that w_i occurs in position i of w
  - Note that a symbol may occur more than once in the same string
  - e.g., for $w = aba$, $w_2 = b$

Alphabets, Languages, Strings, Symbols (4)

Operations and relations on strings

- **Concatenation** for strings x and y yields string xy
  - Concatenation is an associative operation:
    $xyz := (xy)z = x(yz)$
- Eg: for $\Sigma = \{a, b, c\}$, if $x = ab$, $y = bac$ and $z = bba$:
  $xyz = abbacbba$

Alphabets, Languages, Strings, Symbols (5)

More operations and relations on strings

- String v is a **substring** of string w if and only if there are strings x and y such that $w = xvy$
    - If $y = \epsilon$ then $w = xv$ and v is a **suffix** of w
    - If $x = \epsilon$ then $w = vy$ and v is a **prefix** of w
- Eg: if $w = abbacbba$ then
    - $cbba$ is a suffix of w and
    - abb is a prefix of w

Alphabets, Languages, Strings, Symbols (6)

And more operations and relations on strings

- A string w written backwards is denoted w^R and called the **reversal** of w
- Eg: If $w = abbacbba$ then $w^R = abbcabba$

Alphabets, Languages, Strings, Symbols (7)

- A **language** is a set of strings over an alphabet Σ
  - Set operations apply to languages (eg, union, intersection, set difference)
- For a language L over alphabet Σ, its **complement** $\bar{L}$ is $\bar{L} = \Sigma^* - L$ (or $\bar{L} = \Sigma^* \backslash L$)
- Given languages L_1 and L_2 over alphabet Σ, their **concatenation**, denoted $\boldsymbol{L_1 L_2}$ is defined by

$$L_1 L_2 = \{w \in \Sigma^* \mid w = xy \text{ for some } x \in L_1 \text{ and } y \in L_2\}$$

Alphabets, Languages, Strings, Symbols (8)

- The **Kleene star** $\boldsymbol{L^*}$ of a language L is the set of all strings obtained by **concatenating zero or more strings** from L:

$$L^* = \{w \in \Sigma^* \mid w = w_1 w_2 \ldots w_k,\ k \geq 0 \text{ and } w_i \in L \text{ for } 1 \leq i \leq k\}$$

- Given a language L over alphabet Σ, the **closure** $\boldsymbol{L^+}$ of L is $L^+ = LL^*$

    - smallest language that includes L and all strings that are concatenations of strings in L

Didn't we say we wanted to study problems and their solutions?

- A (**decision** or **yes/no**) **problem** is a mapping from a set of **problem instances** to Yes/No (called **yes-instances** and **no-instances**)

- Languages: abstract representation of problems

- For a problem Π, the associated language L_Π is $L_\Pi = \{x \in \Sigma^* \mid x \text{ is a yes-instance of } \Pi\}$

Examples: Yes-No-Problems and their Languages

SORTED SEQUENCE

Input: A list of n comparable elements $e_1, e_2, \ldots, e_n$

Question: Are the elements, as given, in sorted order? That is: is it true that $e_1 \leq e_2 \leq \ldots \leq e_n$?

$L_{\text{SORTED SEQUENCE}} = \{$list of comparable elements l | the elements of l are in sorted order$\}$

Examples: Yes-No-Problems and their Languages

CONNECTED GRAPH

Input: A simple, undirected graph $G = (V, E)$

Question: Is G connected? That is: for any pair of vertices $x, y \in V$, does there exists a path from x to y in G?

$L_{\text{CONNECTED GRAPH}} = \{G = (V, E) \mid G$ is a simple, undirected connected graph$\}$

Examples: Yes-No-Problems and their Languages

SHORT SPANNING TREE

Input: A simple, undirected, edge-weighted graph $G = (V, E)$ where each edge $e \in E$ is assigned a positive integer weight $\mathrm{w}(e)$, an integer k

Question: Does there exist a spanning tree $T = (V, E_T)$ for G where T has weight at most k? That is: T is a tree, $E_T \subseteq E$, and $\sum_{e \in E_T} w(e) \leq k$?

$L_{\text{SHORT SPANNING TREE}} = \{(G = (V, E), k) \mid k$ is a positive integer and G is a simple, undirected, edge-weighted graph has a spanning tree of weight at most $k\}$

Languages / Yes/No problems: How many are there?

- To figure this out, since the set of all languages over an alphabet Σ is defined as the set of all subsets of Σ, first
  - Determine the size of Σ^* for any alphabet Σ

How large is Σ^*?

Claim: Σ^* is countably infinite (and therefore countable)

Proof for $\Sigma = \{0,1\}$:

$\mathbb{N}$	Σ^*
0	ε
1	0
2	1
3	00
4	01
5	10
6	11
…	…

How large is the set of all languages over Σ?

- Recall: Σ^* is countably infinite (and therefore countable)
- and: Languages are subsets of $\Sigma*$
- Then the **set of all languages** over alphabet Σ is the set of all subsets of $\Sigma*$

$$\mathscr{P}(\Sigma*)$$

- For every language L: L is countably infinite or finite
- How large is the set of all languages over alphabet Σ?
  - $|\mathscr{P}(\Sigma*)|$

$|\mathscr{P}(\Sigma^*)| = ?$

- Idea: Show that powerset of any countably infinite set is uncountable
    - This would imply that the powerset of $\mathcal{P}(\Sigma^*)$ is uncountable
- Recall: Any countably infinite set has a bijection with $\mathbb{N}$, that is Σ^* has a bijection with $\mathbb{N}$
- Therefore, if we show that $\mathscr{P}(\mathbb{N})$ is uncountable then we know that $\mathscr{P}(\Sigma^*)$ is uncountable, ie uncountably infinite

$\mathscr{P}(\mathbb{N})$ is uncountable

Proof by contradiction

- **Assume** that $\mathscr{P}(\mathbb{N})$ **is countable.** Then $\mathscr{P}(\mathbb{N})$ is countably infinite

- We list every subset of $\mathbb{N}$ as $S_0, S_1, S_2, \ldots$ s.t.
  - every subset of $\mathbb{N}$ is equal to a subset S_i for some i
- Consider subset $D \subseteq \mathbb{N}$: $D = \{i \in \mathbb{N} \mid i \notin S_i\}$
  - For each $j \in \mathbb{N}$, $j \in D$ if and only if $j \notin S_j$
  - Since $D \subseteq \mathbb{N}$: D is on above list & there is some $j_0 \in \mathbb{N}$ with $S_{j_0} = D$
    - If $j_0 \in D$ then $j_0 \notin S_{j_0} = D$
    - If $j_0 \notin D$ then $j_0 \in S_{j_0} = D$
  - That is: $j_0 \in D$ if and only if $j_0 \notin D$

Goal: determine a subset that should be on the list but is not

Used diagonalization to achieve contradiction

How large is the set of all languages?

- We recap
  - Since $\mathscr{P}(\mathbb{N})$ is uncountably infinite, the powerset of any countably infinite set is uncountable
  - Since the set of all languages is the powerset of Σ^*, the set of all languages is uncountable

So far

- Review, terminology and warmup
- Foundational for upcoming course concepts

Next up: Finite Automata and Regular languages

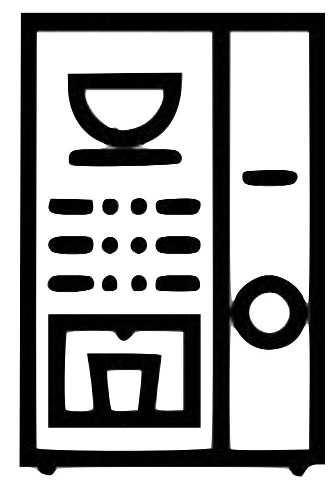
- we begin looking at automata that describe some of those languages

Automata Theory

- **Finite Automata & Regular Languages**
- Pushdown Automata & Context-Free Languages

Finite Automata & Regular Languages

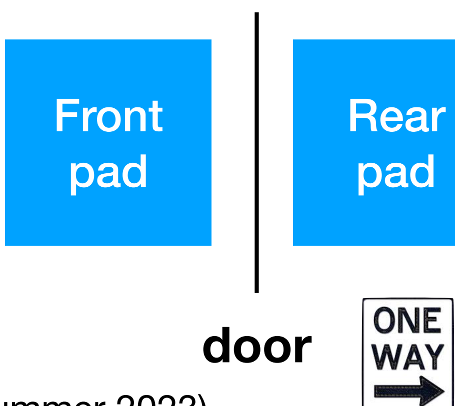
- Understanding **computability** requires
  - Model(s) of computer that capture(s) its computational power
- Most simple model
  - **Finite state machine** or **finite automaton**
  - Model of computation with **finite amount of memory**, independent of problem size



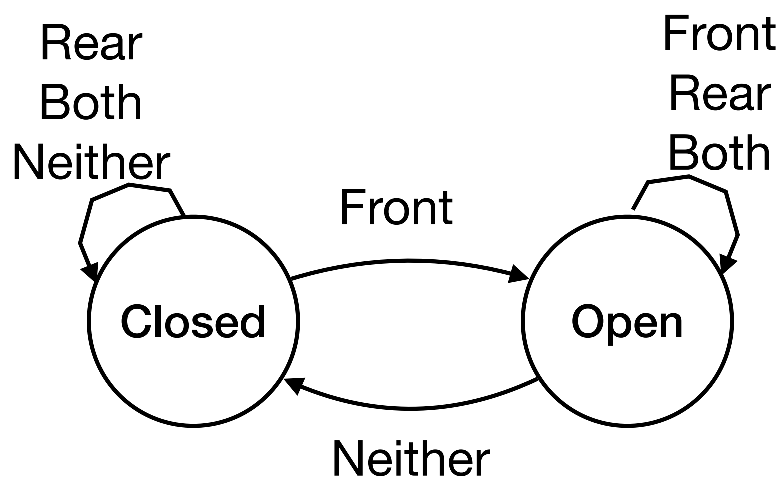
Finite Automata

- Finite automata are all around us! Example: electromechanical devices
  - Controller of an automatic door

Automatic entrance door view from above



Controller



State diagram

Finite Automata in Practice

- Automatic door controller: a finite automaton/computer with just a single bit of memory that records which of the two states the controller is in (closed or open)
- Many other common devices have controllers with somewhat larger memories
- **Elevator controller**
  - state represents floor elevator is on
  - inputs: signals received from the buttons
- **Controllers for various household appliances**
  - dishwashers, electronic thermostats, parts of simple digital watches and simple calculators

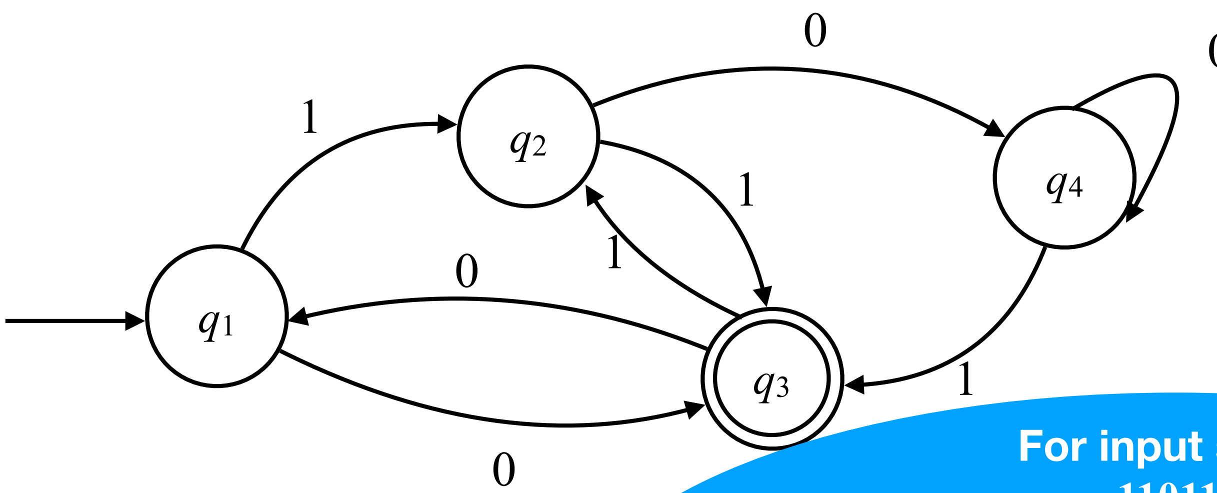
Other Applications of Finite Automata

- Pattern recognition
- Speech recognition
- Optical character recognition
- Compilers
- A (probabilistic) relative of the finite automaton: **Markov chain**

Abstract Description of the Finite Automaton

- **State diagrams**: used to describe finite automata
- Formal Definition: **Deterministic finite automaton**

State Diagram

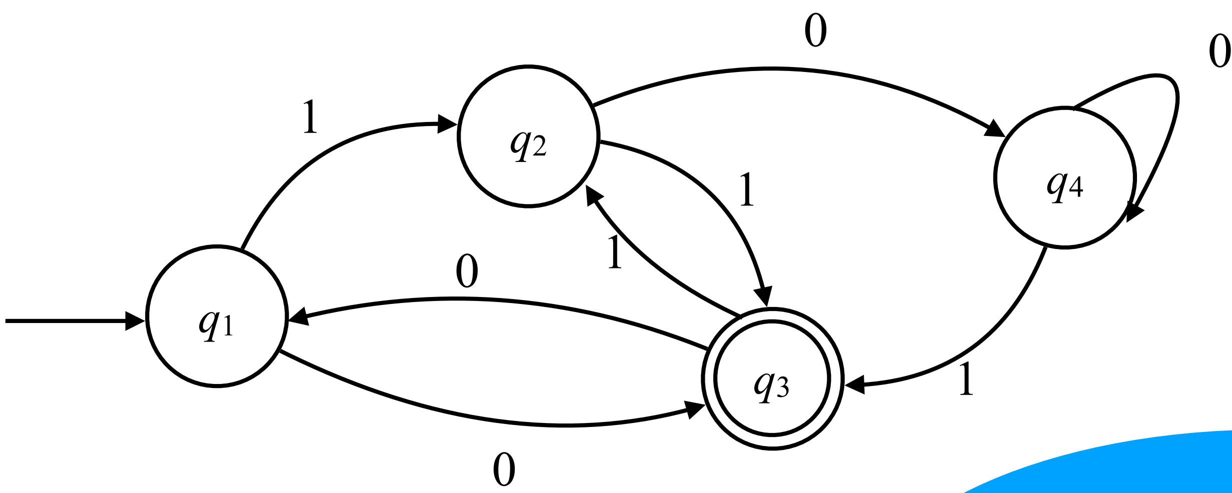


For input string **11011:**
Start in q_1, read 1
Now in q_2, read 1
Now in q_3, read 0
Now in q_1, read 1
Now in q_2, read 1
In q_3

DFA accepts 11011 **since it is in an accept state at end of input string processing**

- States $\{q_1, q_2, q_3, q_4\}$
- Start state q_1
- Accept state q_3
- Transitions: arrows from one state to another (according to received inputs)
- Inputs (labels on transition): symbols from alphabet

State Diagram



- States $\{q_1, q_2, q_3, q_4\}$
- Start state q_1
- Accept state q_3
- Transitions: arrows from one state to another (according to received inputs)

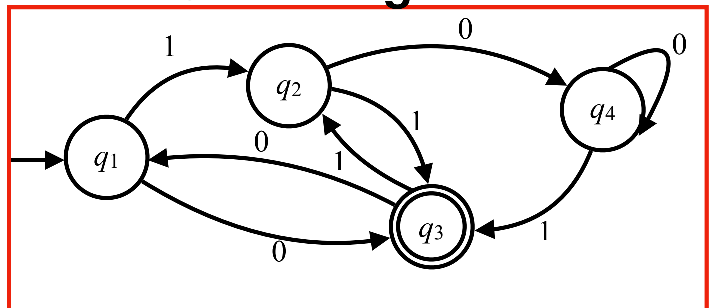
What other strings does the automaton accept?

Formal Definition
Deterministic Finite Automaton

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

1. Q is a finite set called the **states**
2. Σ is a finite set called the **alphabet**
3. Function δ: $Q \times \Sigma \rightarrow Q$ is the **transition function**
4. $q_0 \in Q$ is the **start state**
5. $F \subseteq Q$ is the **set of accept** (or **final**) **states**

State diagram



$(\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_3\})$ with

δ: $Q \times \Sigma \rightarrow Q$ defined by

δ	0	1
q_1	q_3	q_2
q_2	q_4	q_3
q_3	q_1	q_2
q_4	q_4	q_3

Transition table