Lecture 2: Countability and Terminology

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@ uvic.ca

University of Victoria

$\mathbb{R}$ is uncountable (Cantor's Diagonalization)

Proof by contradiction: Assume that the rational numbers $\mathbb{R}$ is countable.

- If $\mathbb{R}$ is countable, then we should be able to enumerate the real numbers **just between 0 and 1**.
- Let the enumeration $(x_1, x_2, x_3, \ldots)$ be written as follows:

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

- $x_n = 0.d_{n1}d_{n2}d_{n3}d_{n4}\ldots$ is the n^{th} real number in the enumeration
- x_n has decimal digits $0.d_{n1}d_{n2}d_{n3}d_{n4}$ (since we are enumerating real numbers between 0 and 1)

$\mathbb{R}$ is uncountable (Cantor's Diagonalization)

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

$x_1 = 0.d_{11}d_{12}d_{13}d_{14} \ldots$ $\boldsymbol{c} \neq x_1$ since the 1^{st} decimal digit is different ($c_1 \neq d_{11}$)

$x_2 = 0.d_{21}d_{22}d_{23}d_{24} \ldots$ $\boldsymbol{c} \neq x_2$ since the 2^{nd} decimal digit is different ($c_2 \neq d_{22}$)

$x_3 = 0.d_{31}d_{32}d_{33}d_{34} \ldots$ $\boldsymbol{c} \neq x_3$ since the 3^{rd} decimal digit is different ($c_3 \neq d_{33}$)

$x_4 = 0.d_{41}d_{42}d_{43}d_{44} \ldots$ $\boldsymbol{c} \neq x_4$ since the 4^{th} decimal digit is different ($c_4 \neq d_{44}$)

$\vdots$

$x_n = 0.d_{n1}d_{n2}d_{n3}d_{n4} \ldots$ $\boldsymbol{c} \neq x_n$ since the n^{th} decimal digit is different ($c_n \neq d_{nn}$)

$\vdots$

- Since $\boldsymbol{c}$ is a number between 0 and 1, it **should be enumerated** in this list
- However, since it **differs from every element**, it cannot be in this list

Clarification on c

- Consider the number $c = 0.c_1c_2c_3c_4 \dots$ where $c_i \neq d_{ii}$ for each i
- For example, suppose the numbers $(x_1, x_2, x_3, \dots)$ are as follows

$$x_1 = 0.\mathbf{4}031\dots$$
$$x_2 = 0.1\mathbf{8}93\dots$$
$$x_3 = 0.53\mathbf{6}7\dots$$
$$\vdots$$

- We define $c = 0.c_1c_2c_3c_4$ such that the digit c_i is something different than the i^{th} digit of x_i

- In the example enumeration above:
    - c_1 can be any number other than 4
    - c_2 can be any number other than 8
    - c_3 can be any number other than 6
    - So, c could be something like 0.597…

Clarification on c

- You may be wondering, if we enumerate the real numbers between 0 and 1 like

$$\begin{aligned} x_1 &= 0.000 \ldots 00 \\ x_2 &= 0.000 \ldots 01 \\ x_3 &= 0.000 \ldots 02 \\ &\vdots \end{aligned}$$

then c must be in the list somewhere.

- Consider if c appears in the list at position k, that is $x_k = c$
- However, c is defined such that digit c_k is different than the k^{th} decimal digit of x_k
- Thus, c can't possibly be in the list anywhere

$\mathbb{R}$ is uncountable (Cantor's Diagonalization)

- The enumerated list $x_1, x_2, x_3, \ldots$ **does not** contain all real numbers between 0 and 1 since it cannot contain c
- So, **we cannot enumerate** all the elements in this subset of $\mathbb{R}$ (real numbers between 0 and 1)
- This is a **contradiction** since we assumed that $\mathbb{R}$ is countable
- Therefore, $\mathbb{R}$ **is uncountable**

Terminology Review: Sets

- **Set**: a collection of distinct **elements** / **members** (unordered, no repeats, can be finite or infinite)
  - $\boldsymbol{S} = \{3, l, 20, \text{green}, \alpha\}$

- Set **membership** / **non-membership**: $\alpha \in \boldsymbol{S}$, $\beta \notin \boldsymbol{S}$

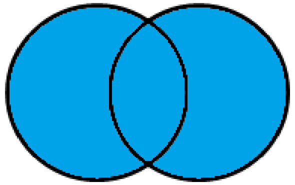
- **Empty set**: Set with no elements
  - $\emptyset$ or $\{\}$

- **Singleton set**: set with exactly one member

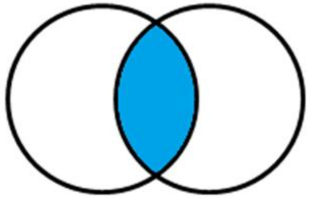
- **Unordered pair**: set with exactly two members

Terminology Review: Set Operations

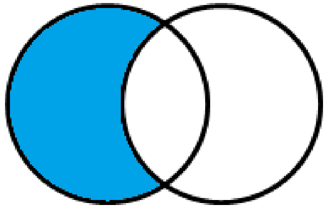
- **Union** of sets $\boldsymbol{A}$ and $\boldsymbol{B}$: $\boldsymbol{A} \cup \boldsymbol{B} = \{\, x \mid x \in \boldsymbol{A} \text{ or } x \in \boldsymbol{B} \,\}$



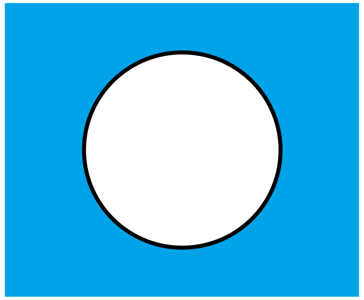
- **Intersection** of sets $\boldsymbol{A}$ and $\boldsymbol{B}$: $\boldsymbol{A} \cap \boldsymbol{B} = \{\, x \mid x \in \boldsymbol{A} \text{ and } x \in \boldsymbol{B} \,\}$



- **Set difference** of sets $\boldsymbol{A}$ and $\boldsymbol{B}$: $\boldsymbol{A} \setminus \boldsymbol{B}$ or $\boldsymbol{A} - \boldsymbol{B} = \{\, x \mid x \in \boldsymbol{A} \text{ and } x \notin \boldsymbol{B} \,\}$



- **Complement** of set $\boldsymbol{A}$: $\overline{\boldsymbol{A}} = \{\, x \mid x \notin \boldsymbol{A} \,\}$



Terminology Review: Powerset

- **Powerset** $\mathcal{P}(A)$ of set A: set of all subsets of A
  - $\mathcal{P}(A) = \{\, S \mid S \subseteq A \,\}$
  - Note that $\emptyset \in \mathcal{P}(A)$ since the empty set is a subset of all sets

- Example: Let $A = \{1, 2, 3\}$. Then $\mathcal{P}(A)$ is

$$\left\{ \begin{matrix} \emptyset, \\ \{1\}, \{2\}, \{3\}, \\ \{1, 2\}, \{1, 3\}, \{2, 3\}, \\ \{1, 2, 3\} \end{matrix} \right\}$$

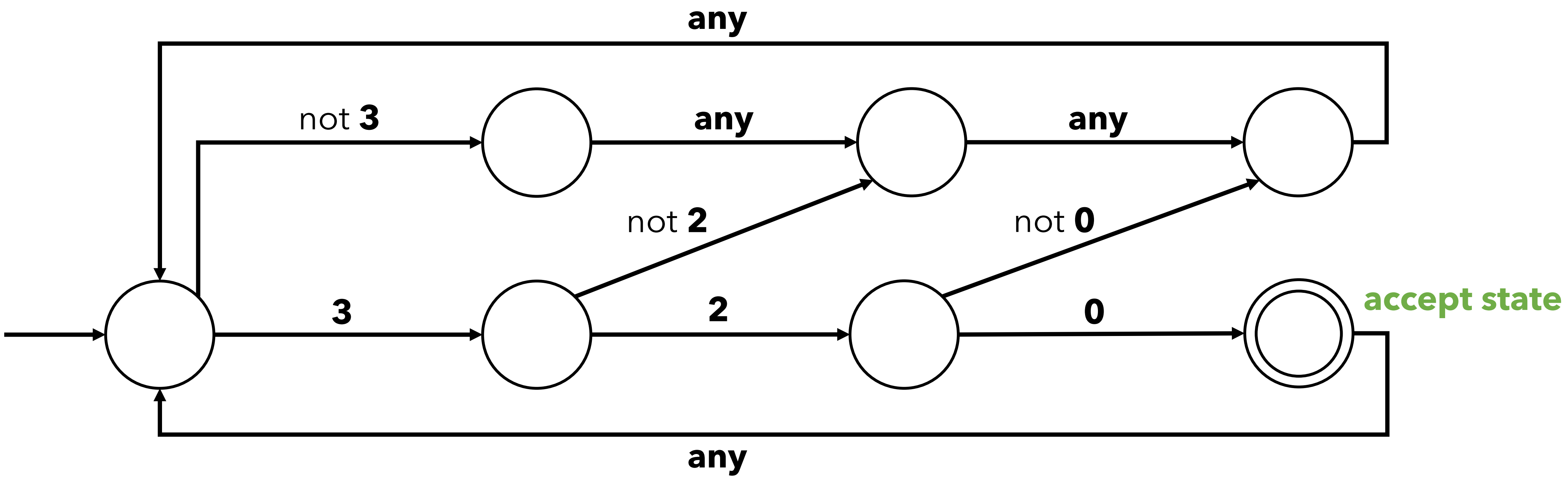
Languages

- In this course, we will be evaluating the **computational power** of different computational models using **languages**
  - "How **complex** of a language can a **model compute** / **represent**?"

- A language in this course is no different than other languages you know:
  - Given an **alphabet**, a language contains **selected strings** created by **symbols** in the alphabet
  - The **English** language: strings containing letters $\{\mathrm{a} - \mathrm{z}\}$ which follow the rules of the language
  - **Java**: strings (text files) containing typed symbols which follow Java syntax

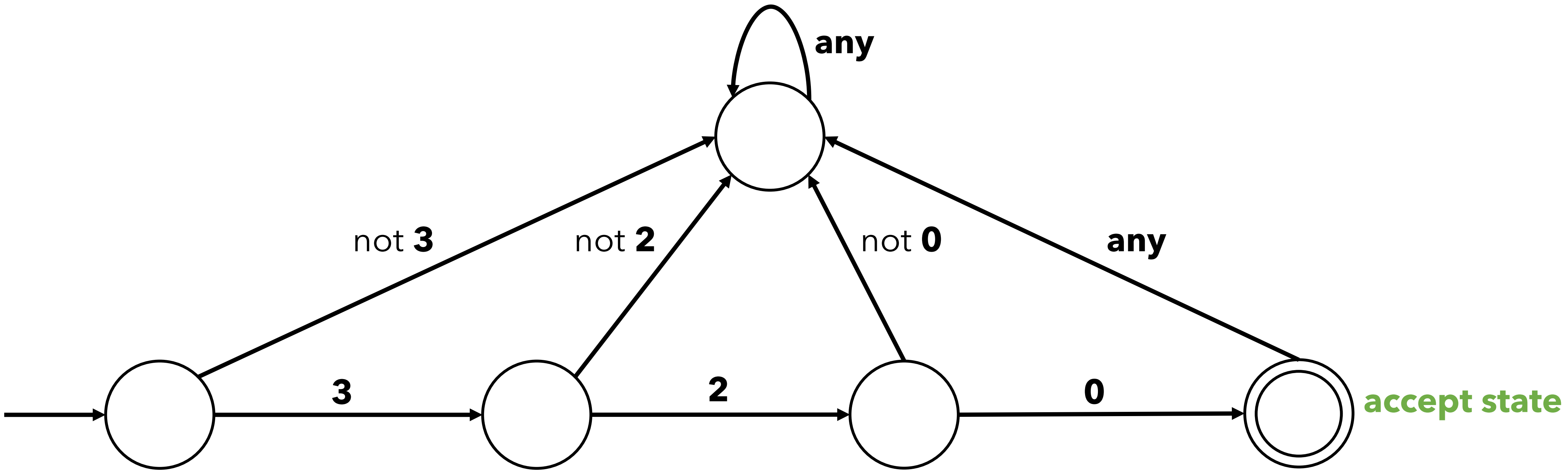
Deterministic Finite Automata

- **DFA state diagram** for the 3-digit passcode game:



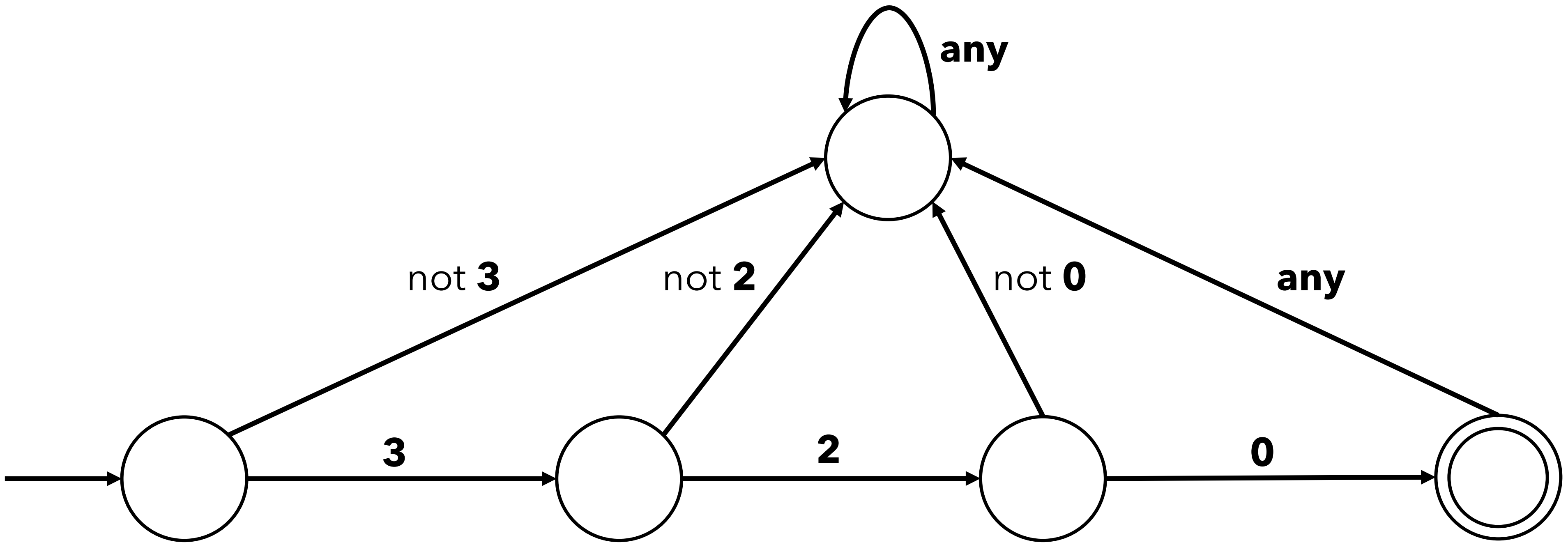
Deterministic Finite Automata

- If we don't display ***** and don't reset the game, we can simplify the DFA to:



- **Accepts** if the input is **320**
- Does **not accept** (lands on a **non-accept state**) if the input is **shorter** than 3 digits, the **wrong** 3 digit code, or **longer** than 3 digits

Languages



- In the passcode game, alphabet is $\{\ \mathbf{0-9}\ \}$ and the language is $\{\ \mathbf{320}\ \}$

- DFA's are powerful enough to represent this language and others
- However, DFA's can't represent the language "**0's followed by the same number of 1s**"

Terminology: Strings

- An **alphabet** Σ is a **finite** set of symbols
  - e.g. binary alphabet $\{0, 1\}$, the Roman / Latin alphabet
- A **string** over an alphabet $\boldsymbol{\Sigma}$ is a finite sequence of symbols from $\boldsymbol{\Sigma}$
  - e.g. 0001 is a string over alphabet $\{0, 1\}$
- The **empty string** ε is the string with no symbols
- The **set of all possible strings** over an alphabet $\boldsymbol{\Sigma}$ is denoted $\boldsymbol{\Sigma}^*$
  - **Important note**: $\boldsymbol{\varepsilon}$ is in $\boldsymbol{\Sigma}^*$ ($\boldsymbol{\varepsilon}$ is a string over any alphabet)
- Example: Let $\Sigma = \{a, b\}$. Then $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa,$ $bab, bba\ bbb, \ldots\}$

Terminology: Strings

- The **length** $|w|$ of a string $\boldsymbol{w} \in \boldsymbol{\Sigma}^*$ is the number of symbols of $\boldsymbol{w}$
  - Length of the empty string: $|\boldsymbol{\varepsilon}| = 0$
  - Let $\boldsymbol{w} = \mathrm{ab}$, $|\boldsymbol{w}| = 2$

- For a string $\boldsymbol{w} \in \boldsymbol{\Sigma}^*$, the symbol in the $\boldsymbol{i^{th}}$ **position** of $\boldsymbol{w}$ is denoted $\boldsymbol{w_i}$
  - Let $\boldsymbol{w} = \mathrm{aba}$, $\boldsymbol{w_2} = b$

Terminology: String Operations and Relations

- The **concatenation** for strings $\boldsymbol{x}$ and $\boldsymbol{y}$ yields string $\boldsymbol{xy}$
  - Let $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, and strings $\boldsymbol{x} = \mathrm{ab}$, $\boldsymbol{y} = \mathrm{bac}$, and $\boldsymbol{z} = \mathrm{bba}$
  - $xyz = \mathrm{abbacbba}$

- A string $\boldsymbol{v}$ is a **substring** of string $\boldsymbol{w}$ if and only if there are strings $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\boldsymbol{w} = \boldsymbol{xvy}$
  - If $\boldsymbol{x} = \boldsymbol{\varepsilon}$, then $\boldsymbol{w} = \boldsymbol{vy}$ and $\boldsymbol{v}$ is a **prefix** of $\boldsymbol{w}$
  - If $\boldsymbol{y} = \boldsymbol{\varepsilon}$, then $\boldsymbol{w} = \boldsymbol{xv}$ and $\boldsymbol{v}$ is a **suffix** of $\boldsymbol{w}$
  - Example: $\boldsymbol{w} = \mathrm{abbacbba}$, then cbba is a suffix of $\boldsymbol{w}$, and abb is a prefix of $\boldsymbol{w}$

- A string $\boldsymbol{w}$ written backwards is denoted $\boldsymbol{w^R}$ and is called the **reversal** of $\boldsymbol{w}$
  - If $\boldsymbol{w} = \mathrm{abc}$, then $\boldsymbol{w^R} = \mathrm{cba}$

Terminology: Languages

- A **language** is a **set of strings** over an alphabet Σ

- Since languages are sets, we can **apply set operations** to languages (union, intersection, etc.) to create new languages

- For a language L over an alphabet Σ, its **complement** $\overline{L}$ is $\overline{L} = \Sigma^* - L$
  - i.e. all strings in Σ^* that aren't in L

- Given languages L_1 and L_2 over alphabet Σ, their **concatenation** denoted L_1L_2 is defined as $L_1L_2 = \{w \in \Sigma^* \mid w = xy \text{ for some } x \in L_1 \text{ and } y \in L_2\}$
  - E.g. Let $L_1 = \{0, 10\}$ and $L_2 = \{0, 11\}$. Then $L_1L_2 = \{00, 011, 100, 1011\}$

Terminology: Languages

- For a language $\boldsymbol{L}$ over alphabet $\boldsymbol{\Sigma}$, the **Kleene star** L^* of $\boldsymbol{L}$ is the set of all strings obtained by **concatenating zero or more strings** from $\boldsymbol{L}$

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

- Example: Let $\boldsymbol{L} = \{0, 10\}$, then $\boldsymbol{L}^* = \{\varepsilon, 0, 10, 00, 010, 100, 1010, \ldots\}$

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

$$\boldsymbol{L}^+ = \boldsymbol{L}\boldsymbol{L}^*$$

- Basically, $\boldsymbol{L}^+$ is $\boldsymbol{L}^*$ but without the "zero strings concatenated" case
- Example: Let $\boldsymbol{L} = \{0, 10\}$, then $\boldsymbol{L}^+ = \{0, 10, 00, 010, 100, 1010, \ldots\}$

Decision Problems

- In this course, the kinds of problems we will be working with are **decision problems**, which are problems with a **yes or no** answer

- Formally, a decision problem is a **mapping** from a set of **problem instances** (inputs) to **yes / no** (yes-instances and no-instances)

- **Examples:**
  - Did the user input the passcode 320?
  - Does the input consist of 0's followed by the same number of 1's?
  - Is the given sequence in sorted order?

Decision Problems Languages

- We can use **languages** as an **abstract representation** of decision problems
- The strings in the language are **yes-instances**

$$\boldsymbol{L} = \{\boldsymbol{x} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{x} \text{ is a yes instance of the problem}\}$$

- **Examples:**
  - Did the user input the passcode 320?

$$\boldsymbol{\Sigma} = \{\mathbf{0} - \mathbf{9}\},\ \boldsymbol{L_1} = \{\boldsymbol{x} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{x} = \mathbf{320}\}$$

  - Does the input consist of 0's followed by the same number of 1's?

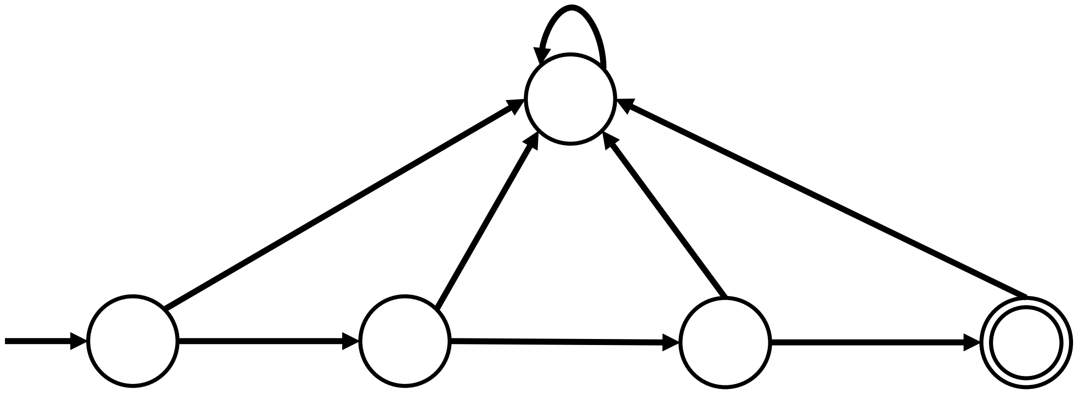
$$\boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}\},\ \boldsymbol{L_2} = \{\boldsymbol{x} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{x} \text{ has form } \mathbf{0}^{\boldsymbol{i}}\mathbf{1}^{\mathbf{i}},\ \boldsymbol{i} \geq \mathbf{0}\}$$

  - Is the given sequence in sorted order?

$$\boldsymbol{L}_{\textbf{SortedSequence}} = \{\text{list of elements } \boldsymbol{l} \mid \text{the elements of } \boldsymbol{l} \text{ are in sorted order}\}$$

Decision Problems and Computational Models

- Recall that we are using **languages** to evaluate the **computational power** of **computational models**
- i.e. can we build an **"algorithm"** in the model to represent the accepted strings in the language



- Then, we are using **languages** as representations of a **decision problem**
- If we can build an **"algorithm"** in the model which **accepts** all yes-instances of a problem language, then the decision problem **can be solved** using the model

Are Decision Problems Enough?

- There are other types of problems that computers solve:
  - **Search problems:** find the desired solution (e.g. find the path between two vertices u and v)
  - **Optimization problems:** maximize or minimize a solution (e.g. find the weight of the minimum spanning tree)

- There is usually a way to form **roughly equivalent decision problems** from other types of problems:
  - e.g. does there exist a path between vertices u and v?
  - e.g. is there a minimum spanning tree of weight $\leq k$?

- Note: the problem **may not be identical** (running time), but it can tell us **if the problem is solvable** on a computational model