COMP 330

Name: Yuhao Wu
ID Number: 260711365

2018-9-05

Relations:

correspondence of elements of two sets:
EXAMPLE: Two sets A and B, $R \subseteq A \times B$ (Cartesian product)
NOTE: R stands for "Relationship"

- not every elements in A needs to be related to anything in B
- A given element in A could be several things in B
- When every element of A is related to exactly one element in B we get a function

Equivalent Relations:

R is an equivalent relation on a set S, $R \subseteq S \times S$

- $\forall s \in S,\ s - R - s \qquad (reflexive)$
- $\forall s, t \in S,\ s - R - t \implies t - R - s \qquad (symmetric)$
- $\forall s, t, p \in S,\ s - R - t, t - R - p \implies s - R - p \qquad (transitive)$

EXAMPLE: $n \equiv m$ (natural numbers) (mod 5) $\implies 32 \equiv 27 (mod \quad 5)$

Definition: Given an equivalence relation R, an equivalence class of an element x

$$\left[x\right] = \left\{y | x - R - y\right\}$$

EXAMPLE: $[2] = \{7, 12, 17, 22, \ldots\} \qquad [2] = [7]$
FACT: Two different equivalence class can't have any elements in common

Partial Order:

Definition: a binary relation $\leq$ (less than or equal to) $<$ (strictly less than)

Axioms:

- $\forall s \quad s \leq s$
- $\forall s$ and $t, \quad s \leq t, t \leq s \implies s = t$
- $\forall s, t,$ and $r, \quad s \leq t$ and $t \leq r \implies s \leq r$

REMARK:

- It might happen that two elements can not be compared.
- If every pairs can be compared, which means that:

$$\forall s \text{ and } t, s \leq t \text{ or } t \leq s \implies \text{(total order)} \qquad \text{[this doesn't always hold]}$$

EXAMPLE 1:
$\{a, b, c\}$
All the subset of $a, b, c \implies \emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}$

$$\left[\text{ the power set (or powerset) of any set } S \text{ is the set of all subsets of } S \right]$$

$\{a\} \subseteq \{a, b\} \subseteq \{a, b, c\}$ [this can be compared] $\{a, b\}$ and $\{a, c\}$ [this can't be compared]

EXAMPLE 2:

$$\textit{natural} \quad \mathbb{N} \qquad \textit{real} \quad \mathbb{R} \qquad \textit{integer} \quad \mathbb{Z}$$

EXAMPLE 3:
$f : \mathbb{R} \to \mathbb{R}$
Define: $f \leq g$ [order of function] $\quad if \quad \forall r \in \mathbb{R}, f(r) \leq g(r)$ [order of numbers (numerical orders)]
This is not total order $\implies$ [$\sin(x)$ and $\cos(x)$]

Well-Founded-Order:

Minimal VS Least:

- Minimal: there is nothing smaller than x (strictly)
- Least: x is smaller than anything else

NOTE: there may be multiple different element to be **Minimal** due to incomparability.

EXAMPLE:

$$U_1 = \{\{a, b\}, \{a, c\}, \{a\}\} \implies \text{least element is } \{a\}$$
$$U_2 = \{\{a, b\}, \{a, c\}, \{a\}, \{b\}\} \implies \text{2 least element is } \{a\}, \{b\}$$

REMARK: Least element is always the minimal

Well-Foundedness: W(set), $(W, \leq)$

Definition: For every non-empty subset U of W, $(U \neq \emptyset \quad \&\& \quad U \subseteq W)$, $\exists u_0 \in U$ such that u_0 is minimal in U.

EXAMPLE:

- $(\mathbb{N}, \leq)$ is well-founded
- $(\mathbb{R}, \leq)$ is not well-founded as you can't find the minimal

NOTE:

- A well-founded total order is called a well-order
- If an order is not well-founded, there are infinitely long strictly descending sequence
- Induction works if your order is well-founded.

$$\text{well-founded} \implies \text{partial order} + \text{ exists minimal}$$
$$\text{well-founded} + \text{total order} \implies \text{well order}$$

Induction:

Suppose that $(S, \leq)$ is a poset, we say that the principle of induction holds for S if for any predicate P , we have

$$\Big[\forall x \in S, (\forall y \in S, y < x \implies P(y)) \implies P(x)\Big] \implies \forall x \in S, \quad P(x)$$

Theorem: An order is inductive if and only if it is well-founded.

Proof. Assume that $(S, \leq)$ is well-founded. Suppose that $P(.)$ is any predicate.
And suppose that $\forall x \in S, (\forall y \in S, y < x \implies P(y)) \implies P(x)$
then we need to prove that $\forall x \in S, P(x)$
Let $U = \Big\{x \in S | \neg P(x)\Big\}$ suppose that there is one element in S such that P(x) is false
Then we need to prove that $U = \emptyset$

Suppose that $U \neq \emptyset$ prove by contradiction
As it is well-founded, there must be a minimal element u_0 in U
$\forall y \leq u_0, y \in S \implies y \notin U$ as u_0 is the minimal element in U, $y \leq u_0$, this is obviously, it is not in U
Then for $\forall y \in S, y \notin U \implies P(x)$
Then, we use the assumption, $\forall x \in S, (\forall y \in S, y < x \implies P(y)) \implies P(x)$,
as we can see that $u_0 \in S, and, y < u_0$, so we can say that $P(u_0)$ holds $\implies u_0 \notin U \implies U = \emptyset$.
Thus, we can conclude that $\forall x \in S, P(x)$

□

Language recognition:

Alphabet: $\Sigma = \{a, b, c\}$ or $\Sigma = \{0, 1\} \implies$ a finite set of symbols
Word: a sequence of letters $\implies$ need to be finite length

EXAMPLE:
Fix $\Sigma = \{a, b\}$, a word can be $a, ab, abb, baba, \ldots$

- An empty word: ϵ
- The collection of all words constructed by Σ is denoted as $\Sigma^* \implies infinity$
- A language $L \subseteq \Sigma^*$

REMARKS:

- $L = \emptyset$ (empty language) $\neq L = \{\epsilon\}$ (a word but an empty word)
- L may be finite or infinite

NOTE:
We need a notation to describe a language, we need algorithm to check if a word is in a certain language.
EXAMPLE: $L = \{w|\text{L has equal number of a and b}\}$

DFA: Deterministic Finite Automata

A DFA is a 4-tuple:

- $S :=$ a finite set of states
- $S_0 \in S :=$ initial state
- δ(transition function) $\quad \delta : S \times \Sigma \to S$
- $F \subseteq S$ accepting states

Definition: A language recognized by **DFA** is called a **Regular language**
Remark: Every state must have an arrow for every letter of Σ

Proposition:

- Every finite language is regular.
- It is possible to have some states that can't be reached.

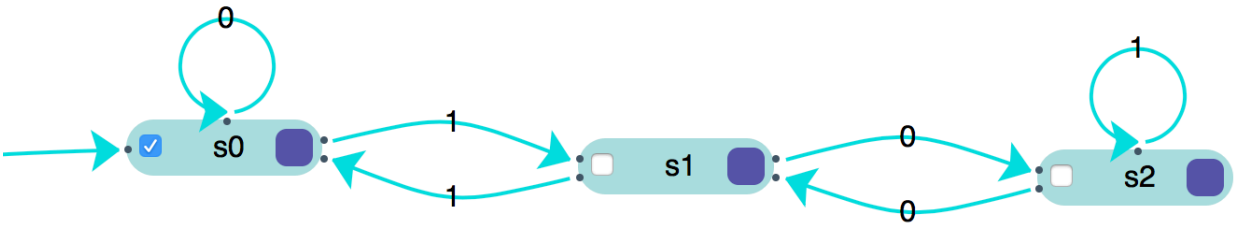
Definition:
$\delta^* : S \times \Sigma \to S$

Inductive Definition:
$\delta^*(s, \epsilon) = s$
$\delta^*(s, aw) = \delta^*(\delta(s, a), w)$

Formal definition of a DFA machine: $L(\mathbb{M}) = \{w \in \Sigma^* | \delta^*(s0, w) \in F\}$

Definition: A language recognized by a DFA is called **Regular Language**

DFA Example:



$\Sigma = \{0, 1\}$ Accept: multiple of 3 in binary number forms

Read and scan numbers
NOTE: DFA won't know what previous number is, it can only read current number and make decisions

according to the current number.
Explanation: There are three states:
S_0 means the remainder is 0, if the next input is 0, which means $x = 2 \times x \implies$ the remainder is 0 as well.
If the next input is 1, which means $x = 2 \times x + 1$, the remainder is 1, so go to state S_1

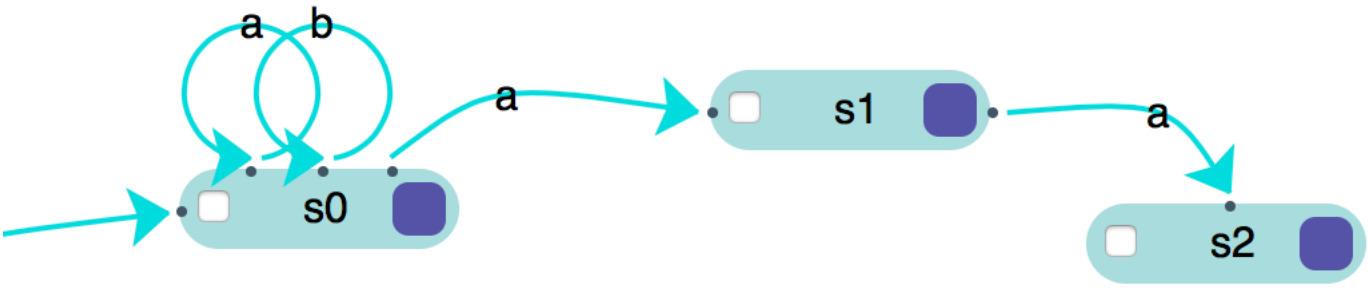
REMARKS:

- Any machine does this job needs at least 3 states.
  There is a unique smallest machine for every language

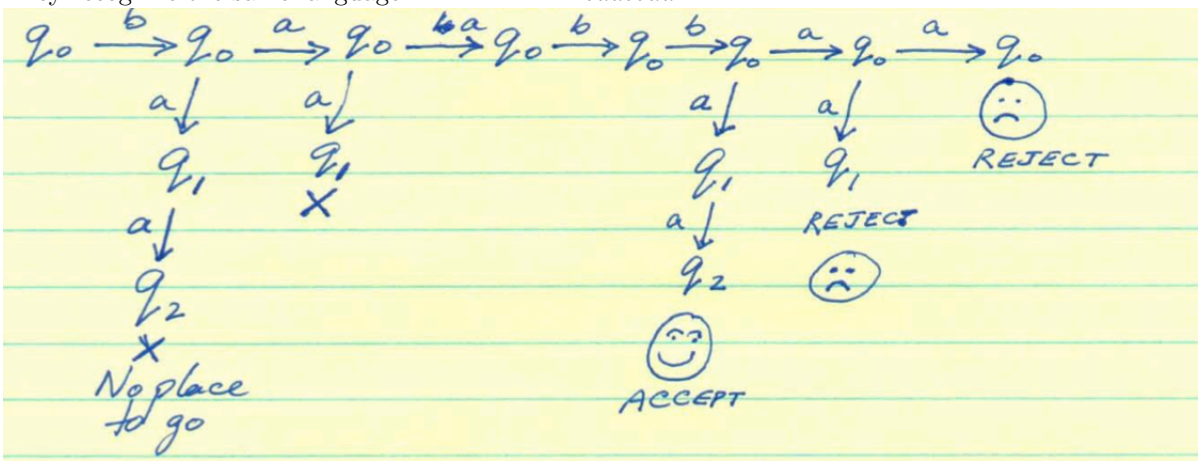
- Suppose that we have a machine which has only two states and can do the job as well.
  We must has one accept state and one reject state.
  Obviously, 100 and 101 belongs to the same state, we add 1, get 1001 and 1011, which leads to different states.

NFA: Nondeterministic Finite Automaton

How to design a machine accepting string ending in "aa".
There is an algorithm which converts any NFA to its equivalent DFA.



They recognize the same language: EXAMPLE: "*baabbaa*"



It doesn't matter that some path going into rejected state as long as there is one path goes to accepted state.

NFA + ϵ: Jump without reading a word:

$\Sigma = \{., +, -, 0, 1, \ldots, 9\}$