Recitation 6

Computability

Review

Defn 1: $\mathcal{P}(S)$ is the set of all subsets of S.

Thm 1: $|\mathcal{P}(S)| = 2^{|S|}$ if S is of finite length. If S is an infinite set then $|\mathcal{P}(S)| > |S|$.

Warm-Up

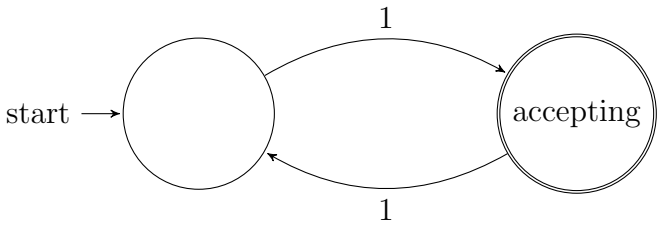
Have you ever wondered what a computer is?

One of the simplest models of computation is a **Deterministic Finite Automata** (DFA), which takes in an input string (usually a **binary string**) and either **accepts** or **rejects**.

Here is how it works:

- Each "circle" is a state with incoming and outgoing "transitions". Start at the state that is labeled "start".
- Read the input string from left to right. For each input character, follow the labeled transitions from the state you are currently in. This may take you to another state.
- Once you have run out of symbols:
  - If you are in a state with a double circle, **accept**.
  - If you are in a state with a single circle, **reject**.

Consider the following DFA, which takes in a string of only 1's.



For example, consider running the DFA on input "1" . You first start in the left state, then see a "1" and move to the right state. Therefore the DFA accepts "1".

a. Does the DFA accept or reject the following strings:

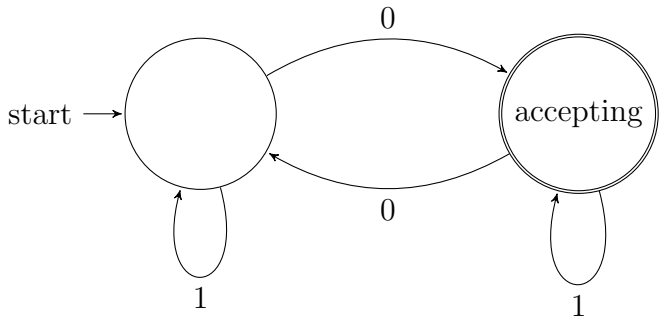
   i. "11"

   ii. "111"

   iii. "11111"

b. What type of strings does this DFA accept? Prove your claim by induction on the length of the input string.

c. From now on we will only be considering DFA's which take **binary strings** as input. Consider the following DFA



Let L be the set of strings accepted by this DFA. Which of the following is correct:

i. $L = \{w \mid w \text{ is a binary string of even length}\}$

ii. $L = \{w \mid w \text{ is a binary string with an odd number of 0's}\}$

iii. $L = \{w \mid w \text{ is a binary string with an odd number of 1's}\}$

d. Let Σ denote the set of 0/1 strings of any length. Is this set infinite?

e. Consider an arbitrary input w to a DFA. Circle all of the correct statements below.

i. $w \subseteq \Sigma$

ii. $w \in \Sigma$

iii. $\{w\} \in \mathcal{P}(\Sigma)$

f. The **language** L of a DFA is the set of strings which the DFA accepts. Circle all of the correct statements below.

i. $L \subseteq \Sigma$

ii. $L \in \Sigma$

iii. $L \in \mathcal{P}(\Sigma)$

g. Design a DFA that accepts a binary string if it ends in a 0. For example, the string "100" is accepted and the string "1101" is rejected.

h. Design a DFA that only takes in strings consisting of 1's and accepts a string only if it is a multiple of 3. (Note that 0 is a multiple of 3)

Section Lesson - The Limit of Computation

Is there anything a computer can't do? (Set Proof)

Our complete model of computation is called a **Turing Machine** after Alan Turing. We call it complete because a Turing Machine can do everything a computer program can do. It is very similar to a DFA and either accepts or rejects a binary string.

A turing machine has a language L of binary strings it accepts.

a. How many languages are there? Write your answer in terms of $|\Sigma|$, $|\mathcal{P}(\Sigma)|$ and explain.

b. Like a computer program, a Turing Machine can be represented as a 0/1 string. How many Turing Machines are there? Write your answer in terms of $|\Sigma|$, $|\mathcal{P}(\Sigma)|$ and explain.

c. What can you conclude from this? Can you build a computer program / Turing Machine for every language?

Side Note - History

From Wikipedia: "Turing is widely considered to be the father of theoretical computer science and artificial intelligence." During the second world war, "Turing played a pivotal role in cracking intercepted coded messages that enabled the Allies to defeat the Nazis in many crucial engagements". Despite this, he was prosecuted as a criminal for homosexual acts and commited suicide in 1952 at age 41.

Proof by Contradiction - *Optional Challenge Problem*

This is confusing and is far beyond the scope of CS22, only read on if you are interested. This is not meant to be part of recitation.

Let P be a computer program. $P(\omega)$ means that we are running the program P on input ω. Both a program and its input can be represented as a 0/1 string. This means that we can run a program on itself, written as $P(P)$.

$P(\omega)$ can either terminate or enter an infinite loop. It would be helpful if we had another program, $HALT$, which would take as input $P(\omega)$ that would tell us if $P(\omega)$ would terminate or not.

You will show that $HALT$ cannot exist with a proof by contradiction.

Assume for sake of contradiction that $HALT$ exists.

Using $HALT$ which we have assumed exists, we can build the program $OPPOSITE$. $OPPOSITE$ does not output anything, it either terminates or loops forever. $OPPOSITE$ takes as input a program P and executes the following steps:

(1) Run $HALT$ on $P(P)$.

(2) If $HALT$ says that $P(P)$ loops forever, then terminate.

(3) If $HALT$ says that $P(P)$ terminates, then loop forever.

Since $OPPOSITE$ takes a computer program as input, we can run it on itself.

Consider running $OPPOSITE$ on $OPPOSITE$.

d. Can $OPPOSITE(OPPOSITE)$ halt?

Hint: If $OPPOSITE(OPPOSITE)$ halts, then what happens in step 2 of $OPPOSITE$? Is this a contradicition?

e. Can $OPPOSITE(OPPOSITE)$ loop forever?

Hint: If $OPPOSITE(OPPOSITE)$ loops forever, then what happens in step 3 of $OPPOSITE$? Is this a contradicition?

f. Have we reached a contradiction? Why? If so we know that the $HALT$ program cannot exist.