CMSC 28100

Introduction to Complexity Theory

Spring 2024
Instructor: William Hoza

Which problems can be solved through computation?

Languages

- A language is a set of strings, all of which are over the same alphabet
- That is, if Σ is an alphabet, then a language over Σ is a set $L \subseteq \Sigma^*$
- Examples:

$$\text{PALINDROMES} = \{w \in \{0,1\}^* : w \text{ is the same forward and backward}\}$$

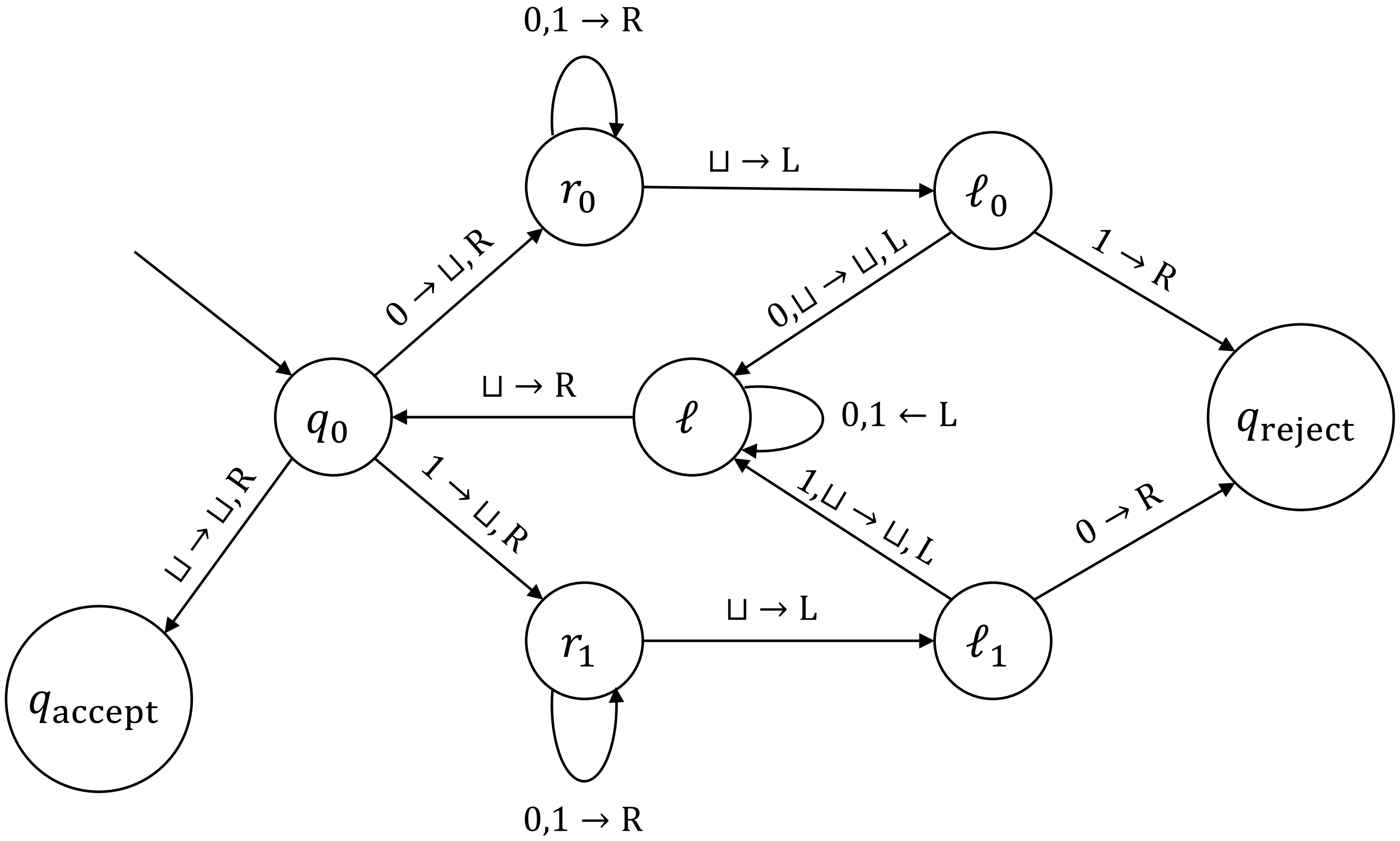
$$\text{BALANCED} = \{w \in \{0,1\}^* : w \text{ has equal numbers of zeroes and ones}\}$$

$$\text{PYTHON} = \text{the set of valid Python programs (no syntax errors)}$$

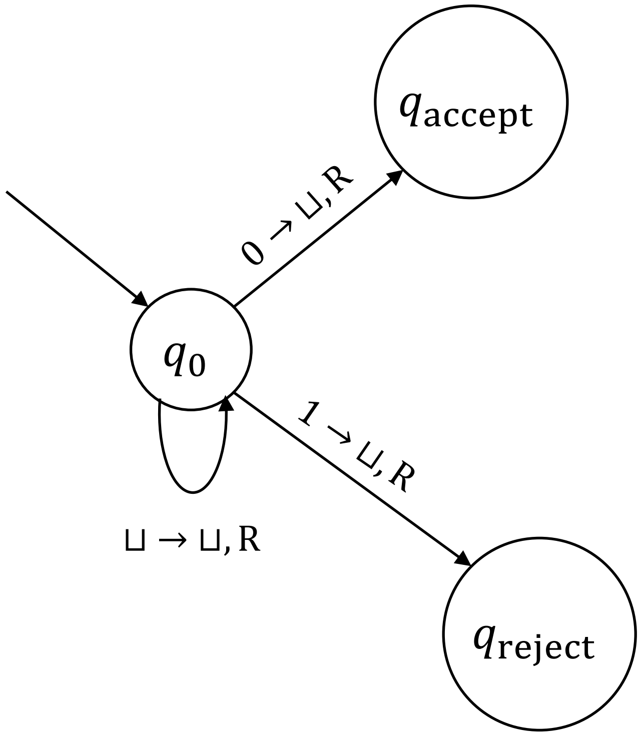
Deciding a language

- Let M be a Turing machine with input alphabet Σ
- Let L be a language over Σ
- Suppose that M accepts every $w \in L$ and M rejects every $w \in \Sigma^* \setminus L$
- In this case, we say that M decides L

Example: A TM that decides PALINDROMES



Example: This TM does not decide any language



Languages as a model of problems

- Each language L represents a computational problem: "Given a string w, determine whether $w \in L$"
  - Given $w \in \{0, 1\}^*$, determine whether w is a palindrome
  - Given a text file, determine whether it is a valid Python program
- "Deciding a language" will be our mathematical model of "solving a problem"

Problems about things other than strings

- **OBJECTION:** “There are many interesting computational problems in which the input is something other than a string.”
- For example, consider the primality testing problem: “Given a positive integer N, determine whether N is prime”
- Does primality testing go beyond the “deciding a language” framework?

Encoding numbers as strings

- **RESPONSE:** If N is a nonnegative integer, we let $\langle N \rangle$ denote the binary encoding of N, i.e., the standard base-2 representation of N
- Example: $\langle 6 \rangle = 110$. Note that $N \in \mathbb{N}$ whereas $\langle N \rangle \in \{0, 1\}^*$
- Primality testing as a language:

$$\text{PRIMES} = \{\langle N \rangle : N \text{ is a prime number}\}$$

Encoding the input as a string

"This is not a pipe."
(1929 painting by René Magritte)

- If we want to give something to a Turing machine, we must first "encode" it as a string
- The same is true of human computation!
- We say, "Given a positive integer, determine whether it is prime," but is it truly possible to "give" someone an abstract concept such as an integer?
- Being pedantic, we could speak more precisely and say, "Given a piece of text, determine whether it represents/encodes a prime number"

Multiple possible encodings

- A problem might be easier or harder depending on how the input is encoded!
- Example: “Given a non-negative integer N, determine whether N is a multiple of ten.”
  - If N is represented in base ten (decimal), the problem is trivial
  - If N is represented in base two (binary), solving the problem requires more effort

Integer divisibility

- Here's another problem: "Given two positive integers, N and M, determine whether N is a multiple of M."

How can we model this problem as a language?

A: $\{\langle N\rangle\langle M\rangle : \exists K,\ N = M \cdot K\}$

B: $\{\langle N\rangle : \exists M,\ \exists K,\ N = M \cdot K\}$

C: $\{\langle N\rangle\#\langle M\rangle : \exists K,\ N = M \cdot K\}$

D: $\{\langle N\rangle\#\langle M\rangle\#\langle K\rangle : N = M \cdot K\}$

Respond at PollEv.com/whoza or text "whoza" to 22333

Encoding a pair of integers as a string

- If N and M are nonnegative integers, then we define

$$\langle N, M \rangle = \langle N \rangle \# \langle M \rangle \in \{0, 1, \#\}^*$$

“Invalid” inputs

- Problem: “Given nonnegative integers N, M, determine whether N is a multiple of M.”

- $L = \{\langle N, M \rangle : N, M \text{ are nonnegative integers and } N \text{ is a multiple of } M\}$

Input	Correct Output	Explanation
100#10	Accept	4 is a multiple of 2
101#11	Reject	5 is not a multiple of 3
1#1#0###	Reject	“Invalid” input

- Convention: We always formulate the language to exclude “invalid” inputs

Encoding graphs as strings

- If G is a graph on N vertices, we let $\langle G \rangle$ denote its adjacency matrix, unraveled into a string, so $\langle G \rangle \in \{0, 1\}^{N^2}$

Encoding other things as strings

- If X is any mathematical object that can be encoded as a string (a number, a graph, a polynomial, a function, …), then we let $\langle X \rangle$ denote some "reasonable" encoding of X as a string
- The specific choice of how to encode X can make a difference, but it usually doesn't make a big difference, provided we choose something reasonable
- If you are unsure how $\langle X \rangle$ should be defined in a particular case, ask!

Beyond decision problems

- “Deciding a language” will be our mathematical model of “solving a problem”
- **OBJECTION:** “There are many interesting problems for which the desired output is something more complicated than a binary yes/no answer.”
- Example: “Sort a given list of integers”
- Example: “Given a graph G, find the largest clique in G”
- (A clique is a set of vertices that are all connected to one another)

Beyond decision problems

- **RESPONSE 1:** We focus on languages for simplicity's sake
- **RESPONSE 2:** In many cases, even if the problem we are interested in is not a decision problem, we can formulate a related language that "captures the essence of" the problem
  - Example: $\text{CLIQUE} = \{\langle G, k \rangle : G \text{ has a clique of size } k\}$
  - More on this later…

Which problems can be solved through computation?

Mathematical models

- Model of “solving a problem:” deciding a language
  - It’s a pretty good model, but admittedly, it does not encompass all possible computational problems
- Model of “computation:” the Turing machine
  - Does this model encompass all possible algorithms?

The Church-Turing Thesis

- Let L be a language

Church-Turing Thesis:

There exists an "algorithm" / "procedure" for figuring out whether a given string is in L if and only if there exists a Turing machine that decides L.

← Intuitive notion

← Mathematically precise notion

Church-Turing Thesis

- The Church-Turing thesis says that the Turing machine model is the "correct" model of arbitrary computation
- The thesis says that the informal concept of an "algorithm" is successfully captured by the rigorous definition of a Turing machine

Are Turing machines too powerful?

- **OBJECTION:** “The Turing machine’s infinite tape is unrealistic!”
- **RESPONSE 1:** If M decides some language, then on any particular input w, M only uses a finite amount of space
- **RESPONSE 2:** We are studying idealized computation