Vocabulary checkpoint: What is the relationship between language, alphabet, encoding, and string?

EECS 376 Discussion 6

Sec 27: Th 5:30-6:30 DOW 1017

IA: Eric Khiu

Slide deck available at course drive/Discussion/Slides/Eric Khiu

Announcement

- Midterm review sessions
  - Daphne: Thursday 2/22 6-8pm LMBE 1130. Topic: Turing Reductions and DP
  - Eric K: Monday 3/4 6-8pm BBB 1670. Topic: Past Exams (will be released soon)
  - Both should be recorded
- Homework 6 deadline extended to Thursday, 2/22
  - Note that we cover content on Monday, 2/19 that is necessary for some problems
- Extra OH next Thursday 2/22- See Piazza Announcement

Computability Recap

- We are interested in "what problems can / can't a computer compute"
- First, we structured what we mean by "problem" by introducing formal languages
- Next, we started to tackle what "computer" means
  - We started by looking at DFAs as computational devices
  - It turns out that DFAs are a little too limited to be a general representation of a computer
  - Now we introduce Turing Machines

Agenda

- Turing Machines
- Decidability
- Counting and Diagonalization

Turing Machines

Course Notes

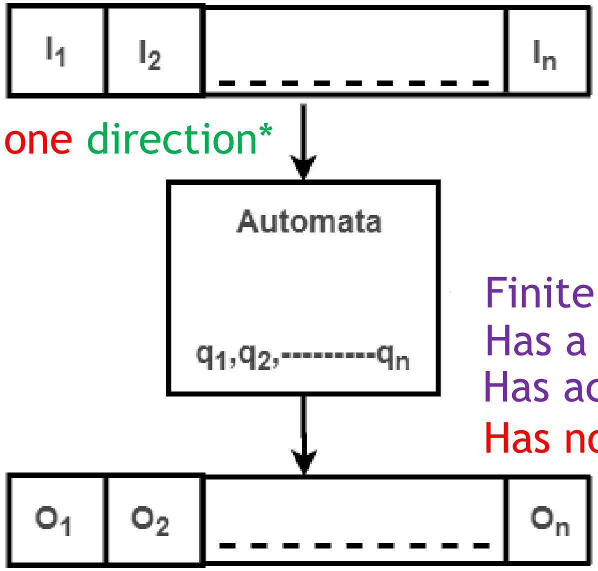
Finite Automata vs Turing Machines

Compare and Contrast:
- Length of tape
- Reading/ writing directions
- Number of states
- Accept/ reject
- Halting condition

Finite Automata

Finite tape with finite input alphabet

Read in one direction*



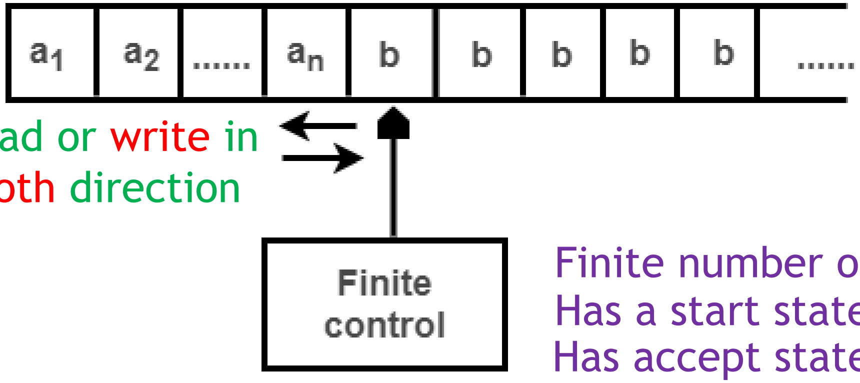
Finite number of states
Has a start state
Has accept states
Has no reject states

Halts after reading all the inputs

Turing Machine

Infinite tape with finite input alphabet and special symbols

Read or write in both direction



Finite number of states
Has a start state
Has accept states
Has reject states

Halts after reaching an accept/ reject state

Otherwise, we say it *loops indefinitely*

*The head of a *two-way automata* can move in both directions, which is out of the scope of this class

Source: https://www.geeksforgeeks.org/difference-between-finite-automata-and-turing-machine/

Definition and Representation

- We define a Turing machine as the 7-tuple $(Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject}, \delta)$
  - Q is a finite set of ***states***
  - $q_0 \in Q$ is the ***initial state***
  - $F = \{q_{\text{accept}}, q_{\text{reject}}\} \subseteq Q$ are the ***final*** (***accept/reject***) states
  - Σ is the ***input alphabet***
  - $\Gamma \supseteq \Sigma \cup \{\perp\}$ is the ***tape alphabet*** ($\perp \notin \Sigma$ is the ***blank symbol***)
  - $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the ***transition function***

Warning: The input string cannot contain the blank symbol $\perp$ and any other symbols in $\Gamma \setminus \Sigma$!

- Turing machines can be represented with state diagrams or pseudocode
  - Turing machines are computationally equivalent to many programming languages
  - It then makes sense to use pseudocode to specify a Turing machine

Demo: turingmachine.io

- https://turingmachine.io/

Decidability

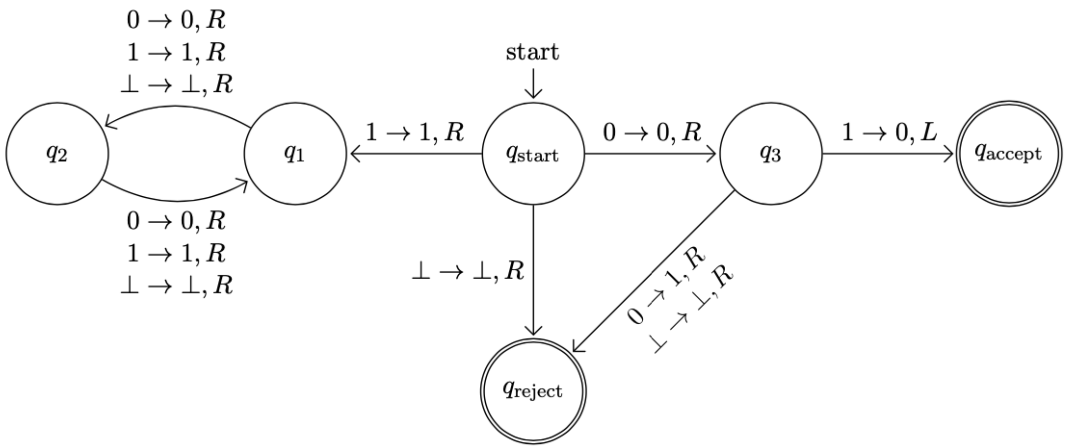
[Course Notes]

Decidability and Turing Machines

- For a language A, we say Turing machine M decides A if:
  - For all $x \in A$, M accepts x
  - For all $x \notin A$, M rejects x
  - And M halts on all input
- Language A is decidable if there exists a TM that decides A
- We call this TM a decider of A

TM State Diagram Practice

Consider the Turing Machine whose state diagram is given below:



- Does this TM accept/ reject/ loops on the following input strings?
  - ε
  - 01
  - 110
- What language over $\Sigma = \{0,1\}$ does this TM decides, if any?
  - None. Observe that if the input strings start with 1, the TM will always loop. In other words, it fails to halt on input of form 1(0|1)*

Proving Decidability

- We have established that a language L is decidable iff there exists some TM that decides L, so proving decidability = construct a TM
- Reminder 1: When we say "give an algorithm", you need to prove the correctness, this applies to TM algorithms too
- Reminder 2: To prove that a TM M decides L, we need to prove
  - For all $x \in L$, M accepts x
  - For all $x \notin L$, M rejects x
  - M halts on all input
- Discuss: Suppose we know some decider exists for some language, can we use it in the decider we want to construct?
  - Yes! Think of it like a *global helper function* that everyone has access to

Decidability Proof Using Known Deciders

- Suppose both S and T are both decidable languages. Prove that $L = S \setminus T$ is decidable
- Since we know that S and T are decidable, we know there exists some TMs, say D_S and D_T that decide S and T respectively.
- We can call those deciders in the TM (decider), D_L we want to build!

Correctness Analysis Draft

- Before we start writing the algorithm, let's start drafting the correctness analysis first (you'll find it useful later)
  - $x \in S \setminus T \Rightarrow \cdots \Rightarrow D_L$ accepts x $\leftarrow$ We want this to happen
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow \cdots \Rightarrow D_L$ accepts x
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow D_S$ accepts x **and** D_T rejects $x \Rightarrow \ldots \Rightarrow D_L$ accepts x
- Otherwise, if
  - $x \notin S \setminus T \Rightarrow \cdots \Rightarrow D_L$ rejects x $\leftarrow$ We want this to happen
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow \cdots \Rightarrow D_L$ rejects x
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow D_S$ rejects x or D_T accepts $x \Rightarrow \cdots \Rightarrow D_L$ rejects x
- We want these two to be the only cases to ensure D_L halts on all input

TM Algorithm

- Construct D_L to make this happens:
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow D_S$ accepts x **and** D_T rejects $x \Rightarrow \ldots \Rightarrow D_L$ accepts x
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow D_S$ rejects x **or** D_T accepts $x \Rightarrow \cdots \Rightarrow D_L$ rejects x

D_L = "On input x:

Run D_S and D_T on x

If $D_S(x)$ accepts **and** $D_T(x)$ rejects **then**

Accept

Reject"

TM Correctness Proof

D_L = "On input x:

Run D_S and D_T on x

If $D_S(x)$ accepts **and** $D_T(x)$ rejects **then**

Accept

Reject

- We just need to complete our proof draft now!
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow D_S$ accepts $x \wedge D_T$ rejects $x \Rightarrow \ldots \Rightarrow D_L$ accepts x
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow D_S$ rejects $x \vee D_T$ accepts $x \Rightarrow \cdots \Rightarrow D_L$ rejects x

TM Correctness Proof

D_L = "On input x:

Run D_S and D_T on x

If $D_S(x)$ accepts and $D_T(x)$ rejects then

Accept

Reject

Now explain what happen here

- We just need to complete our proof draft now!
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow D_S$ accepts $x \wedge D_T$ rejects $x \Rightarrow \ldots \Rightarrow D_L$ accepts x
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow D_S$ rejects $x \vee D_T$ accepts $x \Rightarrow \cdots \Rightarrow D_L$ rejects x

TM Correctness Proof

D_L = "On input x:

Run D_S and D_T on x

If $D_S(x)$ accepts **and** $D_T(x)$ rejects **then**

Accept

Reject

- We just need to complete our proof draft now!
  - $x \in S \setminus T \Rightarrow x \in S \land x \notin T \Rightarrow D_S$ accepts $x \land D_T$ rejects $x \Rightarrow x$ satisfies both conditions to enter the if block, causing D_L to accept $\Rightarrow D_L$ accepts x
  - $x \notin S \setminus T \Rightarrow x \notin S \lor x \in T \Rightarrow D_S$ rejects $x \lor D_T$ accepts $x \Rightarrow \cdots \Rightarrow D_L$ rejects x

TM Correctness Proof

D_L = "On input x:

Run D_S and D_T on x

If $D_S(x)$ accepts **and** $D_T(x)$ rejects **then**

Accept

Reject

- We just need to complete our proof draft now!
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow D_S$ accepts $x \wedge D_T$ rejects $x \Rightarrow x$ satisfies both conditions to enter the if block, causing D_L to accept $\Rightarrow D_L$ accepts x
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow D_S$ rejects $x \vee D_T$ accepts $x \Rightarrow \cdots \Rightarrow D_L$ rejects x

Now explain what happen here

TM Correctness Proof

D_L = "On input x:

Run D_S and D_T on x

If $D_S(x)$ accepts **and** $D_T(x)$ rejects **then**

Accept

Reject

- We just need to complete our proof draft now!
  - $x \in S \setminus T \Rightarrow x \in S \wedge x \notin T \Rightarrow D_S$ accepts $x \wedge D_T$ rejects $x \Rightarrow x$ satisfies both conditions to enter the if block, causing D_L to accept $\Rightarrow D_L$ accepts x
  - $x \notin S \setminus T \Rightarrow x \notin S \vee x \in T \Rightarrow D_S$ rejects $x \Rightarrow x$ satisfies neither conditions to enter the if block, causing D_L to reject $\Rightarrow D_L$ rejects x
  - Additionally, D_L halts on all inputs because if it doesn't enter the if-block, it rejects

Decidability Proof Exercise

Show the following statement is true: For any decidable language L, the language $L' = L \cup \{\varepsilon\}$ is decidable.

```
1: function D_{L'}(x)
2:     if x = ε then
3:         Accept
4:     else if D_L(x) accepts then
5:         Accept
6:     else
7:         Reject
```

$\mathrm{D}_{L'}$ halts on all inputs because D_L is a decider for L.

We have the following implications:

- $x \in L' \implies x \in L \cup \{\varepsilon\} \implies x = \varepsilon \vee x \in L$. If $x = \varepsilon$, then $\mathrm{D}_{L'}$ accepts on lines 2-3. If $x \in L$, then $D_L(x)$ accepts on lines 4-5.
- $x \notin L' \implies x \notin L \cup \{\varepsilon\} \implies x \neq \varepsilon \wedge x \notin L$. Because $x \neq \varepsilon$ and $x \notin L$, x satisfies neither of the conditions which would cause $\mathrm{D}_{L'}$ to accept, so $\mathrm{D}_{L'}$ rejects.

So $\mathrm{D}_{L'}$ decides L'.

Decidability Concept Check 1

Are the following true or false?

(a) Given TM M, there can be more than one distinct language L decided by M

Solution: False, a Turing machine can decide either zero or one languages.

- Deciders are required to halt on all inputs, so any Turing machine that does not halt on some input is not a decider. These machines decide zero languages.
- We now restrict ourselves to deciders. The language of a decider is the set of all (*finite*) strings that machine accepts. Let decider M decide languages L_1 and L_2 For sake of contradiction, suppose $L_1 \neq L_2$. That is, WLOG $\exists x \in L_1 \setminus L_2$. Turing machines are defined as deterministic, so M cannot both accept and reject x, so we've reached a contradiction. L_1 and L_2 must be equivalent, so any decider decides one language.

Decidability Concept Check 2

Are the following true or false?

(b) Given decidable language L, there can be more than one distinct TM M that decides L.

Solution: True. Consider an arbitrary decidable language L and machine that decides it M. We can write a different machine M' that begins by transitioning one cell right, then one cell left, not writing either time, then has an identical transition function to M. Because M' is defined differently, it is a different machine. However, M' will trivially decide L, so M and M' decide the same language. In fact, there are infinite Turing machines for any decidable language.

Recognizability

- For a language A, we say Turing machine M recognizes A if:
  - For all $x \in A,\ M$ accepts x
  - For all $x \notin A,\ M$ does not accept x (this could mean reject or loop!)

Counting and Diagonalization

Course Notes

203 Recap: (Un)countable Infinity

- An infinite set X is countably infinite if you can map each $x \in X$ to a unique natural number (enumerating)
  - More formally, there is a function f such that $f: X \to \mathbb{N}$ is one-to-one (i.e. f is an *injective* function)
- If we cannot write such a function, then the set is uncountably infinite and "strictly larger than" the set of natural numbers

Proving Uncountably Infinite

- We use diagonalization to prove that a set is uncountably infinite
- Ex: Prove that the set of infinite-length binary sequence is uncountably infinite
- Suppose, for the sake of contradiction, that the set is countably infinite, so we can list/ enumerate *every* sequence in an infinite table

Sequence	1st bit	2nd bit	3rd bit	4th bit	5th bit	…
1	0	1	1	0	0	…
2	0	0	0	0	0	…
3	1	0	1	0	1	…
4	1	1	0	1	0	…
⋮						

Proving Uncountably Infinite

- What if we construct a sequence s as follows: 1^{st} bit of s is opposite of 1^{st} bit of sequence 1, 2^{nd} bit of s is opposite of 2^{nd} bit of sequence 2, … i^{th} bit of s is opposite of i^{th} bit of sequence i

Sequence	1^{st} bit	2^{nd} bit	3^{rd} bit	4^{th} bit	5^{th} bit	…
1	0	1	1	0	0	…
2	0	0	0	0	0	…
3	1	0	1	0	1	…
4	1	1	0	1	0	…
⋮						

$s = 1,1,0,0,\ldots$

- By construction, s is an infinite-length binary sequence that is not in the table. Contradiction.

Diagonalization Practice

Let x, y be binary strings of the same length n over $\Sigma = \{0, 1\}$. The *Hamming distance* between x and y, written $d_H(x, y)$, is the number of positions $i \in \{1, 2, \ldots, n\}$ for which $x_i \neq y_i$. For example, $d_H(11100, 10101) = 2$ because the two strings only differ in the second and fifth characters.

Consider an infinite list of infinite binary sequences:

$$
\begin{aligned}
s_1 &= b_{11}b_{12}b_{13}\cdots \\
s_2 &= b_{21}b_{22}b_{23}\cdots \\
s_3 &= b_{31}b_{32}b_{33}\cdots \\
&\vdots
\end{aligned}
$$

Hints:

1. There are infinitely many prime numbers
2. Let p, q be primes and $n, m > 0$. If $p \neq q$, then $p^n \neq q^m$ for all pairs of n, m.

where each $b_{ij} \in \{0, 1\}$. Cantor's diagonalization argument shows that the sequence $\overline{b}_{11}\overline{b}_{22}\overline{b}_{33}\cdots$ has Hamming distance at least 1 from every sequence in the list, where $\overline{b}$ denotes the complement of the bit b.

Construct, with justification, a binary sequences that has *infinite* Hamming distance from each sequence in the list, i.e., it differs from each string in an infinite number of positions.

Diagonalization Practice

Hints:
1. There are infinitely many prime numbers
2. Let p, q be primes and $n, m > 0$. If $p \neq q$, then $p^n \neq q^m$ for all pairs of n, m.

Construct, with justification, a binary sequences that has *infinite* Hamming distance from each sequence in the list, i.e., it differs from each string in an infinite number of positions.

- **Key:** Flip different bits from different sequences, but infinitely many from each
- Let p_k be the k^{th} prime number. Flip all $(p_k)^1, (p_k)^2, \ldots$ bits from the k^{th} sequence
- For example, the first prime number is 2 so we flip the 2nd, 4th, 8th, … bits in the first sequence. Since s_1 is infinite-length, $d_H(s, s_1) = \infty$.

	1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit	9th bit	…
s_1	0	1	1	0	0	1	0	1	1	…
s_2	0	0	0	0	0	1	1	1	1	…
$\vdots$										…
s		0		1				1		…

Diagonalization Practice

Hints:
1. There are infinitely many prime numbers
2. Let p, q be primes and $n, m > 0$. If $p \neq q$, then $p^n \neq q^m$ for all pairs of n, m.

- Let p_k be the k^{th} prime number. Flip all $(p_k)^1, (p_k)^2, \ldots$ bits from the k^{th} sequence
- Next, the second prime number is 3 so we flip the 3rd, 9th, 27th, … bits in the second sequence. Again, since s_2 is infinite-length, $d_H(s, s_2) = \infty$.

	1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit	9th bit	…
s_1	0	1	1	0	0	1	0	1	1	…
s_2	0	0	0	0	0	1	1	1	1	…
$\vdots$										…
s		0	1	1				1	0	…

- By hint 1, we can keep this going because we have infinite primes
- By hint 2, since $p_i \neq p_j \Rightarrow (p_i)^n \neq (p_j)^m$ for all pairs of n, m, there is no collisions in the index of bits flipped
- Therefore, $d_H(s, s_k) = \infty$ for all $k = 1, 2, \ldots$, as desired.

Proving Countably Infinite 1

Show that the set consisting of all the (finite-length) ASCII strings is countable.

- Know: There are 128 ASCII characters which is a finite alphabet, thus the number of strings of length k is 128^k
- **Enumerate:** Shortlex – list the strings by length, then lexicographical order
- Create a list of all strings of each length, then concatenate them together

Length	List	Number of elements	Index of last element
0	[ε]	$128^0 = 1$	1
1	['a', 'b', …]	$128^1 = 128$	129
⋮	⋮	⋮	⋮
k	['aa…a', 'ab…a', …]	128^k	$\sum_{i=0}^{k} 128^k$ ← This is finite!

Therefore, we can map finite-length ASCII strings to natural numbers ⇒ Countably infinite

Proving Countably Infinite 2

Prove that the set of all decidable languages over a given alphabet Σ is countable.

- Know: A TM is represented by a finite-length ASCII string
- Previous: Proven set of all finite-length ASCII strings is countably infinite $\Rightarrow$ set of all TM is countably infinite $\Rightarrow$ we can assign TM to natural numbers
- Know: Each decidable language has at least one unique TM that decides it
  - Previous: No two TMs decide the same language
  - In fact, we have infinitely many TM for one language, but we just need one here
- Map each decidable language *arbitrarily* to one TM that decides it
- Thus, we can map decidable languages through TM to natural numbers
- Therefore, the set of all decidable languages is countably infinite

Existence of Undecidable Languages

- We've shown in lecture the existence of undecidable languages, we now present a counting argument
- Previous: the set of decidable languages is countably infinite
- The set of strings a TM decides is $L(M) \subseteq \Sigma^*$, so the set of all languages is $\mathcal{P}(\Sigma^*)$
  - HW6: Prove power set of countably infinite set is uncountably infinite
  - The set of all languages is uncountably infinite
- Therefore, there must exists some undecidable languages