Midterm Test 1 review, turing machines continued

CSE 105 Week 7 Discussion

Deadlines and Logistics

- Review Test 1 score, schedule attempt 2
- Do review quizzes on PrairieLearn
- HW 5 due 11/19/24 at 5pm

Midterm attempt 1 review

In the area below, create a state diagram of an NFA with **at most 4 states** recognizing the following language over the alphabet $\{0, 1\}$:

$$A = \{w \in \{0,1\}^* \mid w \text{ is odd length string ending in } 1 \text{ or } w \text{ is even length string ending in } 0\}$$

Example strings in this language are: $1, 00, 001$

Example strings not in this language are: $\varepsilon, 0$

In this question, we'll consider general constructions over a fixed alphabet Σ.

Suppose we are given a DFA with set of states Q_M, input alphabet Σ, transition function $\delta_M : Q_M \times \Sigma \to Q_M$, start state q_M, and set of accepting states F_M

$$M = (Q_M, \Sigma, \delta_M, q_M, F_M)$$

and assume that $q_0 \notin Q_M$ is a fresh state.

Define the new NFA

$$N = (Q_M \cup \{q_0\}, \Sigma, \delta_N, q_0, \{q_M\})$$

where

$$\delta_N(\,(q,a)\,) = \begin{cases} \{q' \in Q_M \mid q = \delta_M(\,(q',a)\,)\} & \text{if } q \in Q_M, a \in \Sigma \\ F_M & \text{if } q = q_0, a = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$

- [ ] There is at least one example for the DFA M where $L(M) \neq L(M)^*$ and $L(N) = L(M)^*$.
- [ ] There is at least one example for the DFA M where the number of edges in the state diagram of M equals the number of edges in the state diagram of N.
- [ ] For all choices of the DFA M, the associated NFA N will have strictly more states than M.
- [ ] For all choices of the DFA M, when $L(M) = \Sigma^*$ then $L(N) = \emptyset$.

Definition: A positive integer p is a **pumping length** of a language L over alphabet Σ means that, for each string $s \in \Sigma^*$, if $|s| \geq p$ and $s \in L$, then there are strings x, y, z such that $s = xyz$ and $|y| > 0$, for each $i \geq 0$, $xy^iz \in L$, and $|xy| \leq p$.

In particular, this means that a positive integer p is **not a pumping length** of a language L over alphabet Σ iff

$$\exists s \left(|s| \geq p \wedge s \in L \wedge \forall x \forall y \forall z \left(\; (s = xyz \wedge |y| > 0 \wedge |xy| \leq p\,) \rightarrow \exists i (i \geq 0 \wedge xy^iz \notin L) \right) \right)$$

Select all and only true statements below.

- [ ] A pumping length for $\{0, 1\}$ is $p = 2$
- [ ] A pumping length for $\{0^i0^i \mid i \geq 0\}$ is $p = 2$
- [ ] A pumping length for $\{0^i1^i \mid i \geq 0\}$ is $p = 3$
- [ ] A pumping length for $\{00, 01, 10, 11\}$ is $p = 1$

Select all possible options that apply. ?

Consider an arbitrary alphabet Σ.

- [ ] The class of context-free languages over Σ is closed under complementation.
- [ ] The class of context-free languages over Σ is closed under Kleene star.
- [ ] The class of context-free languages over Σ is closed under union.
- [ ] The class of context-free languages over Σ is closed under set-wise concatenation.

Select all possible options that apply.

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Recall: for any two sets X and Y, we define:

- $X = Y$ means $\forall x(x \in X \leftrightarrow x \in Y)$. In this case, we say X and Y are equal sets.
- $X \neq Y$ means $\exists x(\ (x \in X \land x \notin Y) \lor (x \notin X \land x \in Y)\ )$. In this case, we say X and Y are not equal sets.
- $X \subsetneq Y$ means $\forall x(x \in X \to x \in Y)$ and $X \neq Y$. In this case, we say X is a proper subset of Y.
- $X \supsetneq Y$ means $\forall x(x \in Y \to x \in X)$ and $X \neq Y$. In this case, we say X is a proper superset Y are equal sets.

Select all and only the true sentences.

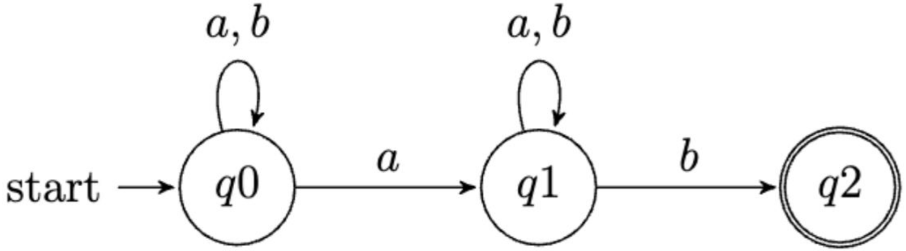
- [ ] The union of any two regular sets is regular.
- [ ] The complement of a nonregular set is nonregular
- [ ] The complement of a regular set is regular.
- [ ] Every proper subset of a nonregular set is nonregular.
- [ ] The union of two nonregular sets is nonregular.

Select all possible options that apply. ?

Recall: for any two sets X and Y, we define:

- $X = Y$ means $\forall x(x \in X \leftrightarrow x \in Y)$
- $X \neq Y$ means $\exists x(\ (x \in X \wedge x \notin Y) \vee (x \notin X \wedge x \in Y)\ )$
- $X \subsetneq Y$ means $\forall x(x \in X \rightarrow x \in Y)$ and $X \neq Y$
- $X \supsetneq Y$ means $\forall x(x \in Y \rightarrow x \in X)$ and $X \neq Y$

Let N be the NFA over the alphabet $\{a, b\}$ with state diagram



$L(N) = \{w \in \{a,b\}^* \mid N \text{ accepts } w\}$

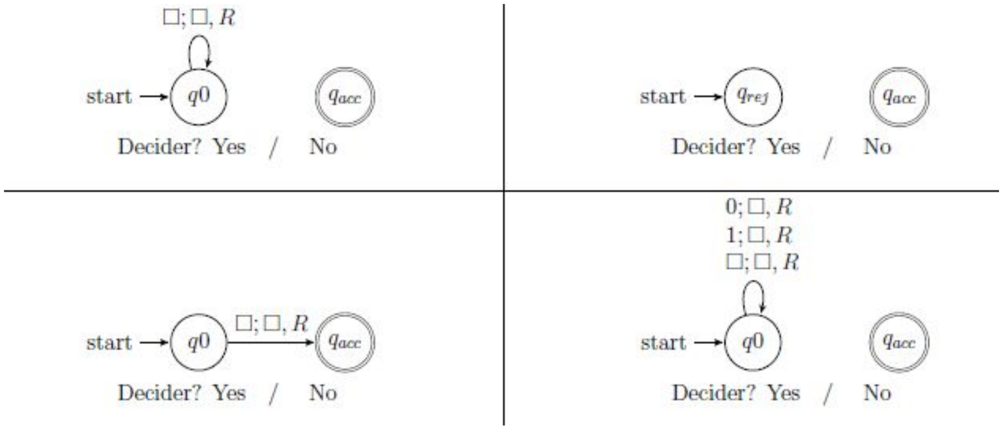
- [ ] $L(N) = L(\ (a \cup b)^* a (a \cup b) b\ )$
- [ ] $L(N) \supsetneq L(\ (a \cup b)^* a (a \cup b) b\ )$
- [ ] $L(N) = L(\ (a \cup b)^* a (a \cup b)^* b\ )$
- [ ] $L(N) \supsetneq L(\ (a \cup b)^* ab (a \cup b)^*\ )$

Select all possible options that apply. ?

Turing-recognizable and Turing-decidable

- Deciders are Turing machines that halt on all inputs; they never loop; they always make a decision to accept or reject
- Call a language Turing-recognizable if some Turing machine recognizes it
- Call a language Turing-decidable if some decider decides it

Toy examples for recap:



Multiple descriptions

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

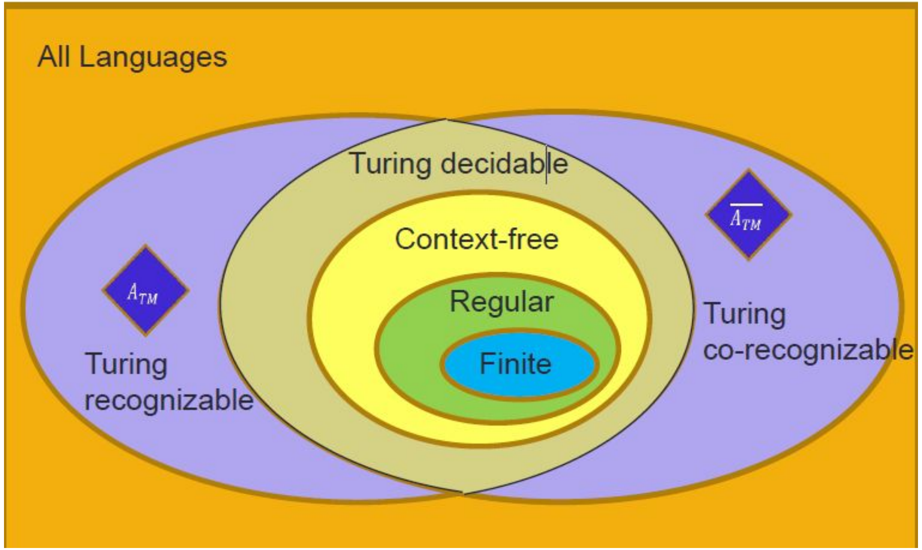
Multiple descriptions

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

Properties of languages

1. Regular
   a. Recognized by a DFA/NFA
   b. Described by a regex
2. Context free
   a. Recognized by a PDA
   b. Generated by a CFG
3. (Turing) Decidable
   a. Can be decided by a Tm
4. (Turing) Recognizable
   a. Can be recognized by a Tm



Algorithm computation

Church-Turing Thesis

Anything that is **computable** is computable with a **Turing machine** because any method of computation using finite time and finite resources will be **equally expressive** to that of a Turing machine.

Vocabulary check

1. Are all decidable languages recognizable?
2. If language A is recognizable and language B is decidable, is |A| > |B|
3. If M is a Turing machine, what is <M>?

Representations of algorithms

To decide these problems, we need to represent the objects of interest as **strings**

For inputs that aren't strings, we have to **encode the object** (represent it as a string) first

To define TM M:

"On input w …

1. ..
2. ..
3. …

Notation:
<O> is the **string** that represents (encodes) the object O
$<O_1, \ldots, O_n>$ is the single string that represents the list of objects $O_1, \ldots, O_n$

Turing Decidable Languages

Recap : Turing decidable languages are closed under complementation

Turing Decidable Languages - Recap

1. If a language is decidable if and only if it is co-recognizable and recognizable.
2. If two languages over a fixed alphabet are turing-decidable, then their union is decidable as well
3. If two languages over a fixed alphabet are turing-recognizable, then their union is recognizable as well

Last Wednesday's "lecture"...

Computational problems:

Acceptance problem		
...for DFA	A_{DFA}	$\{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$
...for NFA	A_{NFA}	$\{\langle B, w\rangle \mid B$ is a NFA that accepts input string $w\}$
...for regular expressions	A_{REX}	$\{\langle R, w\rangle \mid R$ is a regular expression that generates input string $w\}$
...for CFG	A_{CFG}	$\{\langle G, w\rangle \mid G$ is a context-free grammar that generates input string $w\}$
...for PDA	A_{PDA}	$\{\langle B, w\rangle \mid B$ is a PDA that accepts input string $w\}$
Language emptiness testing		
...for DFA	E_{DFA}	$\{\langle A\rangle \mid A$ is a DFA and $L(A) = \emptyset\}$
...for NFA	E_{NFA}	$\{\langle A\rangle \mid A$ is a NFA and $L(A) = \emptyset\}$
...for regular expressions	E_{REX}	$\{\langle R\rangle \mid R$ is a regular expression and $L(R) = \emptyset\}$
...for CFG	E_{CFG}	$\{\langle G\rangle \mid G$ is a context-free grammar and $L(G) = \emptyset\}$
...for PDA	E_{PDA}	$\{\langle A\rangle \mid A$ is a PDA and $L(A) = \emptyset\}$
Language equality testing		
...for DFA	EQ_{DFA}	$\{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$
...for NFA	EQ_{NFA}	$\{\langle A, B\rangle \mid A$ and B are NFAs and $L(A) = L(B)\}$
...for regular expressions	EQ_{REX}	$\{\langle R, R'\rangle \mid R$ and R' are regular expressions and $L(R) = L(R')\}$
...for CFG	EQ_{CFG}	$\{\langle G, G'\rangle \mid G$ and G' are CFGs and $L(G) = L(G')\}$
...for PDA	EQ_{PDA}	$\{\langle A, B\rangle \mid A$ and B are PDAs and $L(A) = L(B)\}$