Exam 3 Practice

Topics

Exam 3 will emphasize material covered since the second exam (as listed below), but questions will also address the “big picture” of the course, including fundamental concepts such as the differences between regular and context-free languages or the models of computation we’ve seen.

You may need to design a DFA, NFA, regular expression, PDA, or CFG as part of a proof or explanation – but you won’t be asked to carry out particular constructions you’ve previously been tested on like converting an NFA to a DFA or to write proofs using the Pumping Lemma.

1 Turing machines

- Formal (tuple) specification
- State transition diagrams
- Using sequences of configurations (instantaneous descriptions) to show the moves of the TM
- Given a Turing machine’s state diagram or specification, be able to describe the machine’s behavior and language.
- Design a Turing machine that recognizes a specific language or behaves in a specific way. Describe it using a state diagram (for simple problems) or pseudo-code.
- Relations between variants of Turing machines, e.g., with a tape infinite in one direction or two, single-tape or multi-tape, deterministic vs nondeterministic
- The universal Turing machine $U$.  

\[ \text{See Sipser p. 176–80} \]
2 Properties of languages

- Definitions of Turing-recognizable (RE) and Turing-decidable (R) languages and how they differ.
- Know the definitions of our standard undecidable languages involving Turing machines, $A_{TM}$ and the Halting Problem.
- Show a language is undecidable because a decider for it would be a self-defeating object; write a proof by contradiction.
- Prove that a language is undecidable using a reduction from another language already known to be undecidable.
- Understand the diagonalization proof that $L_D$ is not Turing-recognizable.

Try the following problems so we can discuss any questions or difficulties you have, either during the review session or during my office hours before the exam. The problems won’t be collected or graded. Collaboration is encouraged; please discuss these problems with classmates to help each other. After the review session, I’ll release example solutions.

You should also review the practice and real problems from the first two exams and/or similar problems from the textbook.
Problem 1: Designing Turing machines

Draw the state-transition diagram for a Turing machine that decides the language

\[ EQUAL = \{ w = w \mid w \in \{a, b\}^* \}, \]

with \( \Sigma = \{a, b, =\} \). Briefly describe how your TM works.
Problem 2: Tracing Turing machines

Consider the Turing machine $M = (\{q_0, q_1, q_{\text{accept}}, q_{\text{reject}}\}, \{0, 1\}, \{0, 1, \square\}, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where $\delta$ consists of the following rules:

\[
\begin{align*}
\delta(q_0, \square) &= (q_{\text{reject}}, \square, R) \\
\delta(q_0, 0) &= (q_0, \square, R) \\
\delta(q_0, 1) &= (q_1, \square, R) \\
\delta(q_1, \square) &= (q_{\text{accept}}, \square, R) \\
\delta(q_1, 0) &= (q_{\text{reject}}, \square, R) \\
\delta(q_1, 1) &= (q_1, \square, R)
\end{align*}
\]

a. Informally but clearly describe the language $L(M)$.

b. Give a string in this language that is of length at least four, and show the sequence of moves of $M$ when run on this input.
Problem 3: Misproving decidability

Explain why the proofs below do not satisfactorily prove the given claims, each of which is false.

a. A nontrivial language is a language that isn’t $\emptyset$ and isn’t $\Sigma^*$. Consider the following language:

$$L = \{(M) \mid M \text{ is a TM and } L(M) \text{ is nontrivial}\}$$

**Claim** $L$ is decidable.

**Proof** Let $M$ be a Turing machine whose behavior is the same as the program given here:

```python
def main(input):
    if len(input) % 2 == 0:
        return True
    else:
        return False
```

Notice that $L(M) \neq \emptyset$ since $M$ accepts the string $\varepsilon$, and that $L(M) \neq \Sigma^*$, since $M$ rejects the string aaa. Moreover, $M$ is a decider since given any input, the machine $M$ will either accept or reject.

This means that $M$ is a decider, $L(M) \neq \emptyset$, and $L(M) \neq \Sigma^*$. Therefore, $L$ is decidable. 

Your answers should identify a fundamental flaw in the proof (for example, by explaining what it actually proves instead of the conjecture), not a minor issue like lack of sufficient detail.
b. CLAIM The language described by the regular expression $a^*b$ is undecidable.

PROOF By contradiction; assume $a^*b$ is decidable. Let $D$ be a decider for it. Consider what happens when we run $D$ on a string of infinitely many as followed by a $b$ and on a string of infinitely many as. Let’s call this first string $x$ and the second string $y$. Since $D$ is a decider, it halts on all inputs, and therefore cannot run for an infinitely long time. Therefore, $D$ must halt before reading the last character of $x$ and the last character of $y$. Because $x$ and $y$ are the same except for their last character, we see that $D$ must have the same behavior when run on $x$ and when run on $y$. If $D$ accepts $x$, then $D$ also accepts $y$, but $y$ is not in the language $a^*b$. Otherwise, $D$ rejects $x$, but $x$ is in the language $a^*b$. Both cases contradict the fact that $D$ is a decider for $a^*b$. We have reached a contradiction, so our assumption must have been wrong. Thus $a^*b$ is undecidable. ■
Problem 4: Self-reference

Use self-reference to prove that the language

\[ L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts string } w \text{ whenever it accepts } w_R \} \]

is undecidable.
Problem 5: Proof by reduction

Let $L = \{\langle M \rangle \mid M$ is a TM that accepts 11 and does not accept 00$\}$. Prove that $L$ is undecidable by giving a reduction from $A_{TM}$. 
Problem 6: Short answer

Read the statements carefully. Circle TRUE or FALSE and briefly justify your answer.

a. TRUE FALSE If language $A$ is a subset of language $B$, and $A$ is Turing-recognizable, then $B$ is Turing-recognizable.

b. TRUE FALSE A nondeterministic TM accepts a string $w$ if and only if every possible computation on the input leads to an accepting state.

c. TRUE FALSE The computational path of a TM on an input $w$ always either halts accepting, halts rejecting, or enters a never-halting loop.
Problem 7: Language classes

Place the following languages in the appropriate part of this (slightly simplified) diagram of the classes of languages:

\[
\begin{align*}
L_a &= \{(M) \mid M \text{ is a TM and } M \text{ accepts } \text{vassar}\} \\
L_b &= \{(M) \mid M \text{ is a TM and } M \text{ rejects } \text{vassar}\} \\
L_c &= \{(M) \mid M \text{ is a TM and } M \text{ loops on } \text{vassar}\} \\
L_d &= \{(M) \mid M \text{ is a TM and } M \text{ has a state named } q_{\text{vassar}}\} \\
L_e &= \{(M) \mid M \text{ is a TM and } M \text{ has an equal number of transitions to } q_{\text{vassar}} \text{ and } q_{\text{acc}}\}
\end{align*}
\]