CSCI 510 **Models of Computation**

HW9: Round and Round

Solution Key

Include your full name*, CS login, and the problem number(s) on each piece of paper you hand in, and please staple your pages together before handing in.*

While collaboration is encouraged in this class, please remember not to take away notes from collaboration sessions other than your scheduled lab section.

Problem 1

In the decision problem SAT*, we are given a list of logical clauses consisting of one or more possibly negated literals, all OR'd together; we say that such a list of clauses is in* SAT *if some assignment of truth values satisfies every single clause. In the optimization version,* MAX SAT*, the goal is to find some assignment of truth values that satisfies the maximum possible number of clauses, even if it is not possible to satisfy every single clause.*

Give a polynomial time 1/2*-approximation algorithm for* MAX SAT*, and show that it satisfies at least* 1/2 *as many clauses in the list as it is possible to satisfy.*

Algorithm: Check if setting all literals to 1 would satisfy at least half of the clauses. If it would, then do that; if it wouldn't, then set all literals to 0.

This works because, if setting all literals to 1 leaves more than half of the clauses unsatisfied, then all those clauses must have only negated literals, so setting all literals to 0 will then satisfy all these clauses. In either case, we get at least half of the total number of clauses, which is at least as good as getting half of the *possible* number of clauses, so we have the 1/2-approximation, as desired.

Problem 2

For each of the following languages, explain why it is decidable, undecidable but Turing recognizable, or unrecognizable.

Note: *"Turing recognizable" is the same as "recursively enumerable."*

a. $L = \{\langle M\rangle \mid \texttt{guitar} \in L(M)\}$.

b. $L = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$.

c. $L = \{\langle D\rangle \mid D \textit{ is an FSM that accepts } \emptyset\}$.

d. $L = \{\langle D\rangle \mid D \textit{ is an FSM that accepts } \Sigma^*\}$.

a. L is recognizable but undecidable. To recognize M, simply simulate M on input `guitar`. If M accepts, then accept, if M rejects, reject, if it loops forever then M does not accept `guitar`, so $M \notin L$, and therefore looping for ever is correct.

   L is undecidable because there are some TMs that accept `guitar`, some that do not, and `guitar` $\in \Sigma^*$. Therefore, by Rice's Theorem, L is undecidable.

b. L is not Turing recognizable. We begin by showing that the language $L_\emptyset$, shown in class to be undecidable, is not Turing recognizable. We use that to show that L is not Turing recognizable.

   We show that $L_\emptyset$ is not Turing recognizable by showing below that its complement $\overline{L_\emptyset} = \{\langle M\rangle \mid \text{accepts some input } w \in \Sigma^* \}$ is recognizable. If $L_\emptyset$ is also recognizable, then $L_\emptyset$ is decidable, which we have shown in class it is not. Thus, $L_\emptyset$ is not Turing recognizable.

   To show that $\overline{L_\emptyset}$ is Turing recognizable, we construct an NTM Q which given $\langle M\rangle$ accepts it if and only if $\langle M\rangle$ is in $\overline{L_\emptyset}$. Q nondeterministically chooses some $w \in \Sigma^*$ and simulates M on w. If M halts and accepts, accept. If M halts and rejects, reject. Finally, if Q loops forever then neither Q nor M accept anything. This machine can accept M. Hence, $\overline{L_\emptyset}$ is recognizable.

   Assume that L is recognizable and produce a contradiction. We reduce $L_\emptyset$ to $L = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$. Given $\langle M\rangle$ we test its membership in $L_\emptyset$ by translating it to $\langle M_1, M_2\rangle$ in which $M_1 = M$ and M_2 is a DTM in $L_\emptyset$ designed to reject all inputs. There clearly is a halting Turing machine that can produce this translation. Thus, $\langle M\rangle \in L_\emptyset$ if and only if $\langle M, M_2\rangle \in L$. If L is recognizable, so is $L_\emptyset$. However, we have shown that $L_\emptyset$ is not recognizable. Thus, the same is true of L.

c. L is decidable. On input $\langle D\rangle$, interpret D as a directed graph (where the states are the vertices and the transitions are the edges) and for each accept state, check if there is a directed path from the start state to an accept state. If there is a directed path for any accept state, reject, otherwise accept.

This will always halt because an FSM has only finitely many states and finitely many transitions, so we can check for all such paths in finite time. It is correct because if there is a directed path from the start state to an accept state, then the input that causes the FSM to follow those transitions will be accepted. If there is no such path, then no input can get to an accept state, so the FSM will reject all inputs and thus it will recognize the language $\emptyset$.

d. L is decidable. First we note that we can assume D is deterministic. This is because we can convert an NFSM into a DFSM with a mechanical process (the power-set construction), so if D is non-deterministic, we can convert it into a DFSM first.

   Now, we use the same idea as in part (c), but check if there is a directed path from the start state to any non-accept state. If yes, reject, otherwise accept. This always halts for the same reason as in part (c). It is correct because the TM is deterministic, so if there is a directed path to a non-accept state, there is an input w on which D will always follow exactly that path and thus D will reject w, so D rejects something. If there is no such path, then D terminates in an accept state on every input, so D accepts Σ^*.

The following questions are lab problems. *Please remember to prepare a solution for your assigned problem before going to your lab section.*

Lab Problem 1

A Turing Machine is **self-terminating** *if it halts when given its own description as input.*

a. *Prove that the language $L_{NST} = \{\langle M\rangle \mid M \text{ is not self-terminating}\}$ is undecidable. You may not use Rice's Theorem.*

b. *Prove that the language L_{NST} is not Turing recognizable. In your proof you should show that the language A_{TM} defined in the last homework is reducible to the complement of L_{NST} (which you may refer to as L_{ST} for simplicity).*

(a) No, L_{NST} is not decidable. Assume, for the sake of contradiction, that it is. Then there exists some decider D_{NST} that decides L_{NST}. We want to find a way to solve the halting problem with this decider. So construct a machine D_H that takes input $\langle M, w\rangle$ and does the following:

1. Constructs a machine M' that completely ignores its input x and simply runs M on w.
2. Runs D_{NST} on M'. If D_{NST} accepts, reject; if it rejects, accept.

Let us examine why this machine decides $HALT_{TM}$:

If M halts on input w, then M' will halt on all inputs. So in particular, M' halts on input $\langle M'\rangle$. So M' is self-terminating, so D_{NST} will reject, so D_H accepts.

If M does not halt on input w, then M' will run forever on all inputs. So, in particular, it will run forever on $\langle M'\rangle$. So M' is not self-terminating, so D_{NST} will accept, so D_H rejects.

Therefore, D_H decides $HALT_{TM}$. But $HALT_{TM}$ is not decidable, so we have a contradiction. Hence, L_{NST} is not decidable.

(b) Proof that L_{NST} is not Turing-recognizable:

First we note that L_{ST} is undecidable because otherwise L_{NST} would be decidable and we already know that is not the case. Next we note that L_{ST} is recognizable. This is because, on input $\langle M\rangle$, we can simply simulate M on input $\langle M\rangle$. If M halts, then it is (by definition) self-terminating, so we accept. If M does not halt, we are stuck in an infinite loop, but M is not self-terminating, so we want to not accept. Therefore L_{ST} is recognizable.

Since L_{ST} is recognizable but undeciable, we know that $\overline{L_{ST}} = L_{NST}$ is not Turing-recognizable.

Lab Problem 2

Explain why each of the following languages is decidable:

a. $L = \{\langle M, w\rangle \mid \textit{when } M \textit{ is run on } w\textit{, it only moves left}\}$

b. $L = \{\langle M, w\rangle \mid \textit{when } M \textit{ is run on } w\textit{, it only moves right}\}$

c. $L = \{\langle M, w\rangle \mid \textit{when } M \textit{ is run on } w\textit{, the head reverses direction at least once}\}$

Note: *When a Turing machine moves left at the start of the tape, it just stays at the start of the tape.*

Hint: *Consider how you would detect an infinite loop, given that you haven't changed direction.*

1. To decide L, construct a table with each entry corresponding to a pair consisting of a state and a tape symbol. Then begin to simulate M on w. As long as you keep moving left, you will stay on the first character of the tape and keep writing it. Every time you see a character in a particular state, mark that pair as visited. Because there are a finite number of symbols and a finite number of states, there are a finite number of $(state, character)$ combinations, and so within a finite amount of time you must either revisit a $(state, character)$ combination, move to the right, or halt. If the former occurs, then you know that you're in an infinite loop, and can halt and accept. If the machine halts, then you can accept, since it has only moved left. And if you ever move right, reject.

2. To decide L, simulate M on w. As soon as you follow a transition to the left, halt and reject. Since you have to keep moving to the right, you must eventually either halt (in which case you accept) or reach blank tape, at which point you will never read any non-blank characters anymore. At that point, start marking the states of M as visited. Because there are a finite number of TM states, you must eventually either halt (in which case you accept) or visit a state for a second time, at which point you know that the TM is in an infinite loop — because you are only reading blanks — and so you can halt and accept.

3. To see that L is decidable, note that we can easily combine the results from parts (a) and (b) of this problem. That is, if we have a TM A which decides part a, and a TM B which decides part b, if we know that M doesn't always go right and M also doesn't always go left, then it must have switched directions. So construct a decider D which takes input M and runs A on M and B on M. If both A and B reject, then accept; else reject.

Lab Problem 3

Consider the following language:

$$L = \{\langle M, k\rangle \mid M \textit{ is a TM that halts on at least } k \textit{ inputs}\}$$

Is this language Turing-recognizable? If so, is it decidable? Prove your answers.

Hint: *You may want to use non-determinism.*

Yes, the language is Turing-recognizable. Consider an NTM that nondeterministically selects k inputs, then runs M on all of those inputs and accepts if they all halt. If there are at least k inputs that halt, then there is at least one accepting branch of the computation, and so we have constructed an NTM that will recognize the language.

However, the language is not decidable. Assume, to the contrary, that it is. Then there exists a decider D for this language. Construct a machine H using D that takes input $\langle M, w\rangle$ and does the following:

Modify M to create M', which first deletes the current input and overwrites it with w, and then runs M. (In other words, no matter the input, M' runs M on w.) Then run D on $\langle M', 1\rangle$, and return the result.

I claim that H solves the halting problem. Note that if M halts on w, then M' halts on all inputs, and so D would accept M' no matter what k is selected. On the other hand, if M does not halt on w, then M' loops on all inputs and halts on none of them, and so D would reject M'. Therefore, H accepts $\langle M, w\rangle$ if and only if M halts on w. In other words, H decides the halting problem. However, the halting problem is undecidable, and so we have a contradiction. Thus, our assumption that L is decidable must be false, making L undecidable.