Name: _____

Student ID: _____

CS181 Winter 2018 - Final

Due Friday, March 16, 11:59 PM

- This exam is open-book and open-notes, but any materials not used in this course are prohibited, including any material found on the internet. **Collaboration is prohibited.** Please avoid temptation by not working on the final while you are in the presence of any other student who has taken or is currently taking CS181. **Be extra careful if you live with or meet regularly with a student of this class.** If you have any questions about the exam, ask the TA or Professor Sahai by email or after class. **Do not ask other students.** You are allowed to use any theorem shown in class or in the textbook, as long as you clearly cite it. Please monitor Piazza for any clarifications. **Do not** post any questions on piazza.
- We suggest that you spend approximately 12 hours (not necessarily contiguous) to take this exam. Start early so that you have time to understand and think about the problems. **The solutions must be submitted on CCLE by 11:59 PM on Friday, March 16.**
- Place your name and UID on every page of your solutions. Retain this cover sheet and the next sheet with the table as the first pages of your solutions. **Please use separate pages for each question. All problems require clear and well-written explanations.**
- There are 4 questions worth a total of 255 points and an extra credit question worth 40 points.
- For each part (except for the extra credit), if you describe a non-trivial approach that you tried using to solve the problem but realized it doesn't work and explain correctly why it doesn't work and then write "I don't know" you will get 20% points for that problem. You will **not** get 20% points for just writing "I don't know". Whether your stated approach was indeed non-trivial is solely at the discretion of the grader.
- 5% extra credit will be awarded to solutions writen in LaTeX.

Please **handwrite** the following honor code agreement and sign and date in the spaces provided.
Honor Code Agreement: I promise and pledge my honor that during the exam period, I did not and will not talk to any person about CS 181 material except for the professor or the TA, nor will I refer to any material except for the class textbook and my own class notes. I will abide by the CS181 Honor Code.

Signature: _____

Date: _____

Question	**Points**	
1		
2		
3		
4		
EC		
Total		

Devin Liu
UID

1a.

Theorem 1a. Prove the pumping lemma for nLRAs.

Proof. Let M be our nLRA with p = $|Q|$ states. Let s be our input string. We can denote the states visited when processing some input of finite length a as the following:

$$q_0 q_1 ... q_{\mathrm{a}} \tag{1}$$

During this set of state transitions, we will assume that there are no loops meaning each state visited is unique. We can denote the pattern of directions taken by the robot so far as w.

We continually recycle our input s since we want to generate an infinite pattern from a finite input. Thus, we will have an infinite number of state transitions. However, we only have a finite number of states in our nLRA. This means that there must exist a state which is visited multiple times and there exists a loop somewhere in the nLRA. Let state q_{a} be the first state of such a loop. We denote the pattern of directions taken by the robot while this loop is occurring as z. Thus, the pattern of directions taken by the robot overall is of the form wzzzzz... □

1b.

Theorem 1b. Show that the language $L = \{(<< R >>, s)|$ For all n $\in \mathbb{N}$, R(s) eventually visits the n-th cell on the infinite tape$\}$ is decidable.

Proof. To show that our language L is decidable, we have to show that it will halt on any input x. We can define a TM M which decides L. We can construct our TM M as follows:

M = "On input ($<< R >>$,s), where $<< R >>$ is a nLRA and s is an input:
1. Simulate $<< R >>$ on input s.
2. After visiting a state q_{r} for a second time, remember that state and mark the cell that $<< R >>$ is at in the tape.

3a. If $<< R >>$ ends on a cell to the right of the marked cell after visiting q_{r} a third time, then accept.

3b. If $<< R >>$ ends on or to the left of the cell after visiting q_{r} a third time, then reject.

Explanation: For this problem, we can use the pumping lemma for nLRAs. We know that the pattern of directions taken by the robot is of the form wzzzzz.... Since we know that the pattern loops, we can halt once we hit the z portion of the pattern if we don't move to the right of the marked cell because it will never advance further down the tape. However if it does move to the right of the marked cell, it will keep moving right on subsequent iterations of the loop. Thus, it will eventually reach the n-th cell. □

Devin Liu
UID

2a.

Theorem 2a. HALT = $\{(< M >, x)|$ M halts on input x$\}$ is Libra decidable.

Proof. To show that HALT is Libra decidable, we need to show that HALT will either accept or reject x.

We construct a Libra Machine L which decides HALT as follows:
L = "On input ($< M >$,x) where $< M >$ is a TM and x is an input:

1. Construct a TM N which loops forever so that L(N) = $\emptyset$.
2. Load $< M >$ and $< N >$ into the two machine tapes of L.
3. Perform language equality query on $< M >$ and $< N >$.
   3a. If L enters no state, then accept.
   3b. If L enters yes state, then go to step 4.
4. Construct a TM M' where M' is the complement of M, i.e. it inverts the accepting and rejecting states of M.
5. Load $< M' >$ and $< N >$ into the two machine tapes of L.
6. Perform language equality query on $< M' >$ and $< N >$.
   6a. If L enters no state, then accept.
   6b. If L enters yes state, then reject.

Explanation: I constructed $< N >$ to test whether $< M >$ would accept or reject anything. Because L(N) = $\emptyset$, I can gather information about what $< M >$ accepts and rejects. If L(M) $\neq$ L(N), then $< M >$ accepts some input and thus halts. If L(M) = L(N), then $< M >$ accepts nothing. If this is the case, we then have to check if $< M >$ rejects nothing as well. If it does, then $< M >$ doesn't halt. We construct $< M' >$ which is the complement of M. We then check to see what L(M') is using the language equality query. If L(M') $\neq$ L(N), then this means M rejects everything. Thus, $< M >$ halts. However if L(M') = L(N), then M does not accept or reject anything. Thus M must loop and M $\notin$ HALT. Thus, HALT is Libra decidable. □

2b.

Theorem 2b. MINIMAL = $\{< M > |$ M is a minimal turing machine$\}$ is Libra decidable.

Proof. To show that MINIMAL is Libra decidable, we have to show that if $< M > \in$ MINIMAL, then the Libra Machine will accept. Otherwise, it will reject. In no circumstances should our Libra Machine loop.

We construct our Libra Machine L as follows:
L = "On input $< M >$ where $< M >$ is a TM:

1. Assume that some TM E enumerates MINIMAL from shortest to longest description, that is it enumerates the set of all minimal TMs.

2. Load $< M >$ and the first TM N from E onto the machine tapes of L.
Remove N from the enumeration. Load the descriptions of $< M >$ and $< N >$ onto the regular tape.
3. Perform language equality query on $< M >$ and $< N >$.
   3a. If L enters yes state, then go to step 4.
   3b. If L enters no state, then go back to step 2.
4. Compare the lengths of the descriptions of $< M >$ and $< N >$ on the regular tape.
   4a. If |Description of M| > |Description of N|, then reject.
   4b. Otherwise, accept.

Explanation: L uses an enumeration of minimal TMs to determine whether M is minimal or not. L grabs the first TM N in the enumeration and compares it to M. If L(M) = L(N), then we check if the description of N is shorter than the description of M. If it is, then M is not minimal and we reject. Otherwise, M is minimal and we accept. If L(M) $\neq$ L(N), then we get the next minimal TM from the enumeration and repeat the process. L will not loop because the enumeration of minimal TMs must contain a TM N such that L(M) = L(N) (otherwise, it is not a complete enumeration). Thus, we will eventually find such a TM N in the enumeration. As a result, MINIMAL is Libra decidable. □

2c.

Proof. An example of a language that is not Libra recognizable is as follows:

$$MAXIMUM = \{< M > | \text{M is a maximum turing machine}\} \tag{2}$$

MAXIMUM is not Libra recognizable because our Libra Machine L has to compare infinitely many TMs to M due to the fact that there is no bound on the sizes of descriptions of TMs. As a result, L is never sure if M $\in$ MAXIMUM and can never accept or reject anything. We can prove that there are an infinite number of TMs using Cantor's diagonalization. If we have an enumeration of m TMs such as (M_1,M_2...M_m), we can always apply diagonalization to obtain a new TM M_{m+1} that is guaranteed (by diagonalization) to be distinct from any TM in the enumeration. We can then append M_{m+1} to the enumeration and apply diagonalization to the resulting enumeration to get another TM. We can repeat this process forever. Thus, there are infinitely many TMs. As a result, there are infinitely many TMs for L to compare with M. Thus, L can never accept or reject anything. Since it cannot accept or reject anything, it is not Libra recognizable. □

2d.

Proof. An example of a language that is Libra recognizable but not Libra decidable is as follows:

$$FINITE = \{< M > ||L(M)| = n \text{ where n is a natural number}\} \tag{3}$$

Devin Liu
UID

FINITE is Libra undecidable because it cannot handle the case where $|L(M)| = \infty$. If $|L(M)| = \infty$, then our Libra Machine L will loop. However, FINITE is Libra recognizable since looping is allowed in terms of recognizability. To demonstrate this idea, I will build a Libra Machine L as follows:

L = "On input $< M >$ where $< M >$ is a TM:

1. Assume that some TM E enumerates the set of all TMs N such that $|L(N)|$ is finite.
2. Load $< M >$ and the first TM N from E onto the machine tapes of L.
   Remove N from the enumeration.
3. Perform language equality query on $< M >$ and $< N >$.
   3a. If L enters yes state, then accept.
   3b. If L enters no state, go to step 2.

Explanation: Since E does not contain any TMs N such that $|L(N)|$ is infinite, if $|L(M)| = \infty$, then L will loop forever. This fact makes FINITE Libra undecidable. However, looping is permitted in Libra recognizable languages. Thus, this language is Libra recognizable, but not Libra decidable. □

Devin Liu
UID

3a.

Theorem 3a. DISAGREE is undecidable.

Proof. Assume that DISAGREE is decidable. Then, there exists a decider D for DISAGREE. Let us construct a TM M which takes x as its input as follows:

Construct M(x):
Let z = $< M >$ by the recursion theorem
Let N = TM which accepts Σ^*
Run D($< N >$,z)
Case Accept: accept x
Case Reject: reject x

Explanation: If the decider D accepts, then that means it thinks that there exists an x such that x $\in$ L(M_1) but x $\notin$ L(M_2). In this case, we accept all inputs x. Since L(M) and L(N) are both Σ^*, this is a contradiction. If on the other hand D rejects, then it thinks that there doesn't exist an x such that x $\in$ L(M_1) but x $\notin$ L(M_2). In this case, we reject all inputs x. L(N) remains Σ^*, but L(M) is now $\emptyset$. This is clearly a contradiction. Thus, we have proved both cases are contradicting and DISAGREE is undecidable. □

3b.

Theorem 3b. DISAGREE is unrecognizable.

Proof. To prove some language is unrecognizable, we have to find a contradiction for the accept, reject and loop cases. Luckily for us, we can take our solution from 3a and modify it since we have already dealt with the accept and reject cases.

Assume that DISAGREE is recognizable. Then, there exists a recognizer R for DISAGREE. Let us construct a TM M which takes x as its input as follows:

Construct M(x):
Let z = $< M >$ by the recursion theorem
Let N = TM which accepts Σ^*
Run R($< N >$,z)
Case Accept: accept x
Case Reject: reject x
Case Loop: do nothing

Explanation: If the recognizer R accepts, then that means it thinks that there exists an x such that x $\in$ L(M_1) but x $\notin$ L(M_2). In this case, we accept all inputs x. Since L(M) and L(N) are both Σ^*, this is a contradiction. If on the other hand R rejects, then it thinks that there doesn't exist an x such that x $\in$ L(M_1) but x $\notin$ L(M_2). In this case, we reject all inputs x. L(N) remains Σ^*, but L(M) is now $\emptyset$. This is clearly a contradiction. If R loops, then it comes to the same conclusion as the reject case (it thinks that there doesn't exist an x such that x $\in$ L(M_1) but x $\notin$ L(M_2)). However, L(N) is Σ^* while L(M) is $\emptyset$. L(M) is $\emptyset$ because R never gives up control since it is looping which means M cannot accept anything. Thus,

there is a contradiction in the looping case. Now that there are contradictions in all cases, we can conclude that DISAGREE is unrecognizable. □

3c.

Theorem 3c. FIB = {$< M >$ | M is a fibonacci enumerator} is recognizable.

Proof. To prove that FIB is recognizable, I have to construct a TM N which accepts on valid inputs and rejects or loops on invalid inputs. In this case, a valid input would be a fibonacci number n and an invalid input would be everything else.

We construct our TM N as follows:
N = "On input n, where n is a number:
1. If n = 1, accept. Otherwise, load both of the tapes of the TM with an initial value of 1 in binary.
2. Add the values of both tapes and overwrite the first tape with the result r in binary. Look at r.
   2a. If r = n, accept.
   2b. If r != n, go to step 3.
3. Add the values of both tapes and overwrite the second tape with the result s in binary. Look at s.
   3a. If s = n, accept.
   3b. If s != n, go to step 2.

Explanation: It is trivial to prove that the TM N will eventually write n to a tape if n is a fibonacci number. Thus, it will always accept if n is a fibonacci number. If n is not a fibonacci number, it will loop forever between steps 2 and 3 since it will have skipped over the non-fibonacci number. Since the values written in the tape are constantly increasing, there is no chance the machine ever returns to a lower number and thus our machine loops on an input of a non-fibonacci number. Thus, FIB is recognizable. □

3d.

Theorem 3d. The set {$< M >$ | there is some x such that $(< M >, x) \in SMALLANDPICKY$} is infinite where SMALLANDPICKY = {$(< M >, x)$| M is a minimal Turing machine where L(M) = {x}}.

Proof. SMALLANDPICKY is the set of minimal turing machines that accept a language with only one member. To show that the set of minimal TMs ∈ SMALLANDPICKY is infinite, we can use Cantor's diagonalization. Using diagonalization, we can prove that there are an infinite number of languages which consists of only one member. Once we prove this, we get the proof that the set of TMs ∈ SMALLANDPICKY is infinite for free.

Devin Liu
UID

To prove there are an infinite number of languages with only one member, I will use induction.

Base case: Our initial set of languages is {}. There is a language L_1 which consists of only one member. It must be different from our set of languages since it has no languages yet.

Induction case: Assume that we have a set of m languages and each language in the set consists of only one member like so:

$$(L_1 L_2 ... L_m) \tag{4}$$

Using Cantor's diagonalization on the set, we can obtain a language L_{m+1}. L_{m+1} is different from all the languages in the set due to diagonalization. To obtain another unique language, simply append L_{m+1} to the original set and apply diagonalization to the resulting set. Repeating this process forever will give us an infinite number of languages. Thus, the set of languages which consists of only one member turns out to be infinite. As a result, the set of minimal TMs $\in$ SMALLANDPICKY is infinite. □

3e.

Theorem 3e. SMALLANDPICKY is unrecognizable.

Proof. Suppose that SMALLANDPICKY is recognizable. Then there exists a recognizer R for SMALLANDPICKY. Let us construct M.

Construct M(x) given R:
    Let M be a minimal TM.
    Let z = $< M >$ via the recursion theorem.
    If x = ϵ, then accept.
    Run R(z)
        Case Loop: do nothing
        Case Accept: accept x
        Case Reject: reject x

Explanation: If the recognizer R loops, then it thinks that M $\notin$ SMALLANDPICKY. However, it is minimal and only accepts one input, ϵ. This is a contradiction. If R accepts, then it thinks that M $\in$ SMALLANDPICKY. However, it accepts everything and L(M) = Σ^*. This is a contradiction. If R rejects, then it thinks that M $\notin$ SMALLANDPICKY. However, M is minimal and L(M) = $\{\epsilon\}$. This is a contradiction. We have contradictions in all the cases and thus SMALLANDPICKY is unrecognizable. □

Devin Liu
UID

4a.

Theorem 4a. $L_P = \{< M > \mid P(L(M)) = 1\}$ is unrecognizable where $P(\emptyset) = 1$.

Proof. Before we begin the proof, we need to establish a definition.
Def: A nontrivial language property p: languages $\rightarrow \{0,1\}$.

1. $\exists$ L_{Yes} such that $P(L_{Yes})=1$ & $\exists$ TM M_{Yes} such that $L(M_{Yes})=L_{Yes}$
2. $\exists$ L_{No} such that $P(L_{No})=0$ & $\exists$ TM M_{No} such that $L(M_{No})=L_{No}$

Suppose L_P is recognizable. Then there exists a recognizer R for L_P.
Construct M(x):
    Let z = $< M >$
    Run R(z)
        Case Accept: Return M_{No}(x)
        Case Reject: Return M_{Yes}(x)
        Case Loop: Do nothing

Explanation: If R accepts, then it thinks that $P(L(M)) = 1$. We return M_{No}(x) so that $L(M) = L(M_{No}) = L_{No} \rightarrow P(L_{No}) = 0$. This is a contradiction. If R rejects, then it thinks that $P(L(M)) \neq 1$. We return M_{Yes}(x) so that $L(M) = L(M_{Yes}) = L_{Yes} \rightarrow P(L_{Yes}) = 1$. This is a contradiction. If R loops, then it thinks that $M \notin L_P$ and $P(L(M)) \neq 1$. We can't return anything since we don't have control. Thus, $L(M) = L(\emptyset) = \emptyset$. However as we defined in our theorem, $P(\emptyset) = 1$. Thus, we have reached a contradiction. With contradictions in all cases, we have proved that L_P is unrecognizable. □

4b.

Theorem 4b. $L_P = \{(< M_1 >,< M_2 >) \mid P(L(M_1),L(M_2)) = 1\}$ is undecidable.

Proof. Suppose L_P is decidable. Then there exists a decider D for L_P.
Construct M(x):
    Let z = $< M >$
    Let $< N >$ be an arbitrary TM.
    Run D($< N >$,z)
        Case Accept: Return M_{No}(x)
        Case Reject: Return M_{Yes}(x)

Explanation: If D accepts, then it thinks that $P(L(N),L(M_2)) = 1$. We return M_{No}(x) so that $L(M) = L(M_{No}) \rightarrow P(L(N),L(M_{No})) = 0$. This is a contradiction. If D rejects, then it thinks that $P(L(N),L(M_2)) \neq 1$. We return M_{Yes}(x) so that $L(M) = L(M_{Yes}) \rightarrow P(L(N),L(M_{Yes})) = 1$. This is a contradiction. Since we have contradictions in all cases, we have proven that L_P is undecidable. □

Devin Liu
UID

4c.

Theorem 4c. $L_P = \{(< M_1 >,< M_2 >) \mid P(L(M_1),L(M_2)) = 1\}$ is undecidable where $P(L(M_1),L(M_2)) = 1$ and $P(L(M'_1),L(M'_2)) = 0$.

Proof. Suppose L_P is decidable. Then there exists a decider D for L_P.
Construct M(x):
    Let z = $< M >$
    Let $< M_1 > = < M'_1 > = < N >$ be an arbitrary TM.
    Run D($< N >$,z)
        Case Accept: Return $M'_2(x)$
        Case Reject: Return $M_2(x)$

Explanation: For this problem, we modify our proof from 4b. In order to reach a contradiction, we set $< M_1 > = < M'_1 > = < N >$. This allows us to not have to worry about the extra variable in the problem. Once we have done this, we can proceed with the proof the same way we did in 4b.

If D accepts, then it thinks that $P(L(N),L(M_2)) = 1$. We return $M'_2(x)$ so that $L(M) = L(M'_2) \rightarrow P(L(N),L(M'_2)) = 0$. This is a contradiction. If D rejects, then it thinks that $P(L(N),L(M_2)) \neq 1$. We return $M_2(x)$ so that $L(M) = L(M_2) \rightarrow P(L(N),L(M_2)) = 1$. This is a contradiction. Since we have contradictions in all cases, we have proven that L_P is undecidable. □