CS3331 - Assignment 4 - 2018

Decidable and Semi-Decidable Languages II

Due: Monday, Dec 3, 2018 (Latest to submit: Thursday, Dec 6)

1. (20pt) Is the set SD closed under intersection? Prove your answer. Clear English description of any Turing machines is sufficient. (That is, you don't have to effectively build the machine, instead explain how the machine behaves.)

   Yes. Given a Turing machine M_1 to decide L_1 and a TM M_2 that decides L_2 we construct a TM M_3 that decides $L_1 \cap L_2$ as follows. We use three tapes. Tape one will be the working tape, tape two will store the codes of M_1 and M_2, and tape three will be temporary storage. Given an input string w on tape one, copy w onto tape three. Then simulate the running of M_1 on w. If M_1 rejects, M_3 will reject. Otherwise M_1 accepts, and so clear tape one, copy w back to tape one from tape three and simulate the running of M_2 on w. If M_2 rejects, M_3 will reject. Otherwise M_2 accepts, so M_3 accepts.

2. (20pt) If L_1 and L_3 are in D and $L_1 \subseteq L_2 \subseteq L_3$, what can we say about whether L_2 is in D?

   Nothing (L_2 may be in D or it may not). Note that if $L_1 = \varnothing$ and $L_3 = \Sigma^*$, the L_1 and L_3 are regular, and hence decidable, so then every possible language $L_2 \in P(\Sigma^*)$ satisfies the relation $L_1 \subseteq L_2 \subseteq L_3$. Hence, L_2, could be regular, context-free but not regular, decidable but not context-free, semidecidable but not decidable or not semidecidable.

3. (50pt) For each of the following languages, prove, without using Rice's Theorem, whether it is (i) in D or (ii) in SD but not in D.

   (a) $L_1 = \{< M >: \texttt{a} \in L(M) \text{ and } \texttt{b} \in L(M)\}$.

   SD/D: The following algorithm semidecides L:

   Run M on `a`. If M rejects, then loop. Otherwise M accepts, and so run M on `b`. If M accepts, then accept. Otherwise loop.

   This procedure will only accept if M accepts both `a` and `b`. Otherwise it will loop.

   Proof not in D: R is a reduction from $H = \{< M, w >:$ TM M halts on $w\}$ to L_1, defined as follows:

   $R(< M, w >) =$

   1. Construct the description of $M\#(x)$ that, on input x, operates as follows:
   - **1.1 Erase the tape.**
   - **1.2 Write w on the tape.**
   - **1.3 Run M.**
   - **1.4 Accept.**

   2. Return $< M\# >$.

If $Oracle$ exists and decides L_1, then $C = Oracle(R(< M, w >))$ decides H:

- **R can be implemented as a Turing machine.**
- **C is correct: $M\#$ accepts everything or nothing depending on whether M halts on w. So:**
  - **$< M, w > \in H$: M halts on w so $M\#$ accepts everything and thus accepts `a` and `b`, so $Oracle$ accepts.**
  - **$< M, w > \notin H$: M doesn't halt on w so $M\#$ doesn't halt and thus accepts nothing and so does not accept `a` and `b`. $Oracle$ rejects.**

But no machine to decide H can exist, so neither does $Oracle$.

(b) $L_2 = \{< M, w >:$ M, on input w, never writes on its tape$\}$.

D. If $M = (K, \Sigma, \Gamma, \delta, s, H)$ cannot write on its tape, it is effectively an FSM. More importantly, observe that there are $|K|$ states. If M moves off its input onto the blank region, it can count up to $|K|$ blanks past the end of w by keeping the count in its state. If it moves farther than that off the end of w (without writing anything), it is in a loop and it will continue to move without writing, since it must, a second time in state q, reading a blank, do the same thing it did the first time it was in state q reading a blank. So we need consider only $|w| + 2 \cdot |K|$ positions for M's read-write head. Let $max = |K| \cdot (|w| + 2 \cdot |K|)$. If M has not written on its tape after max steps, then it is in a loop and it will just keep doing the same thing over and over. So the following algorithm decides L_2:

1. Run M on w for max steps or until M halts or writes on its tape.

- **If M halted without writing, halt and accept.**
- **If M wrote on its tape, halt and reject.**
- **Otherwise, halt and accept (since it is just going to loop and continue not to write on its tape).**

(c) $L_3 = \{< M, n >:$ M accepts at least one string of length $> n\}$.

SD/D: The following algorithm semidecides L_3:

Run M on the strings in Σ^* of length greater than n in lexicographic order, interleaving the computations (use dovetailing). As soon as one such computation has accepted, halt and accept.

Proof not in D: R is a reduction from $H = \{< M, w >:$ TM M halts on $w\}$ to L_3, defined as follows:

$R(< M, w >) =$

1. Construct the description of $M\#(x)$ that, on input x, operates as follows:

1.1 Erase the tape.

1.2 Write w on the tape.

1.3 Run M.

1.4 Accept.

2. Return $< M\#, 1 >$.

If *Oracle* **exists and decides** L_3**, then** $C = Oracle(R(< M, w >))$ **decides** H**:**

- R **can be implemented as a Turing machine.**
- C **is correct:** $M\#$ **accepts everything or nothing depending on whether** M **halts on** w**. So:**
  - $< M, w > \in H$**:** M **halts on** w **so** $M\#$ **accepts everything and thus accepts a string of length greater than 1, so** $< M\#, 1 > \in L_3$ **and** *Oracle* **accepts.**
  - $< M, w > \notin H$**:** M **doesn't halt on** w **so** $M\#$ **doesn't halt and thus accepts nothing and so does not accept any strings of length greater than 1, so** $< M\#, 1 > \notin L_3$ **and** *Oracle* **rejects.**

But no machine to decide H **can exist, so neither does** *Oracle*.

(d) $L_4 = \{< M >: L(M) \neq L(M_0)$ for any other TM $M_0\}$

D: For any language L**, if a TM** M **accepts** L **then there are infinitely many other Turing machines that accept the same language (by adding inaccessible states). Thus,** $L_4 = \varnothing$**, and is hence regular and therefore decidable.**

(e) $L_5 = \{< M, w >:$ TM M accepts w and rejects $w_R\}$.

SD/D: The following algorithm semidecides L_5**:**

Run M **on** w**. If it accepts, run** M **on** w^R**. If it rejects, accept. In all other cases, loop.**

Proof not in D: R **is a reduction from** $H = \{< M, w >:$ **TM** M **halts on** $w\}$ **to** L_5**, defined as follows:**

$R(< M, w >) =$

1. Construct the description of $M\#(x)$ **that, on input** x**, operates as follows:**

1.1 Save x**.**

1.2 Erase the tape.

1.3 Write w **on the tape.**

1.4 Run M**.**

1.5 If $x =$ `ba`**, then reject.**

1.6 Else accept.

2. Return $< M\#,$ `ab` $>$.

If *Oracle* **exists and decides** L_5**, then** $C = Oracle(R(< M, w >))$ **decides** H**:**

- R **can be implemented as a Turing machine.**
- C **is correct:**
  - $< M, w > \in H$**:** M **halts on** w **so** $M\#$ **always makes it to step 1.5.** $M\#$ **accepts** `ab` **and rejects** `ba`**, so** $< M\#,$ `ab` $> \in L_5$ **so** *Oracle* **accepts.**
  - $< M, w > \notin H$**:** M **doesn't halt on** w **so** $M\#$ **accepts nothing and so** $< M\#,$ `ab` $> \notin L_5$**,**

so *Oracle* **rejects.**

But no machine to decide H **can exist, so neither does** *Oracle***.**

4. (10pt) Prove using reduction from $\neg H = \{< M >:$ TM M does not halt on input $w\}$ that the language

$$H_{\neg \text{ANY}} = \{< M >: \text{there does not exist a string on which TM } M \text{ halts}\}$$

is not in SD. (Hint: see pp. 476-77 in the textbook).

Let R **be a reduction from** $\neg H = \{< M, w >:$ **TM** M **does not halt on** $w\}$ **to** $H_{\neg \mathbf{ANY}}$**, defined as follows:**

$R(< M, w >) =$

1. Construct the description of $M\#(x)$ **that, on input** x**, operates as follows:**
   1.1 Erase the tape.
   1.2 Write w**.**
   1.3 Run M **on** w**.**
2. Return $< M\# >$

If *Oracle* **exists and semidecides** $H_{\neg \mathbf{ANY}}$**, then** R **semidecides** $\neg H$**:**

- $< M, w > \in \neg H$**:** M **does not halt on** w**. So** $M\#$ **gets stuck in step 1.3. It does not halt on any input string, so** $< M\# > \in H_{\neg \mathbf{ANY}}$**. Thus,** *Oracle* **accepts.**
- $< M, w > \notin \neg H$**:** M **halts on** w**. So,** $M\#$ **halts after step 1.3. It halts on every input string, so** $< M\# > \notin H_{\neg \mathbf{ANY}}$**. So** *Oracle* **does not accept.**

But no machine to semidecide $\neg H$ **can exist, so neither does** *Oracle***.**