

COMP 481: Automata, Formal Languages, and Computability
Spring 2008
Solutions to Homework Assignment #9

1. I used R and D interchangeably to denote the set of recursive (decidable) languages, and use RE and SD interchangeably to denote the set of recursively enumerable (semi-decidable) languages.
2. I make use of the facts that $A_{TM} = \{\langle M, w \rangle : M \text{ accepts } w\}$ is in SD but not in D, and that $\overline{A_{TM}} = \{\langle M, w \rangle : M \text{ does not accept } w\}$ is not in SD.
3. I write τ for a reduction function.

1. (a) A 3-tape TM M that computes the sum of two numbers a and b , represented in binary, works as follows (a and b are both on the first tape when M starts):
 - i. If either a or b equals 0, M halts.
 - ii. M copies b to the second tape, positions the reading heads on the first and second tape to point to the rightmost letters in both a and b , and enters a state of ‘no carry’.
 - iii. If in a ‘no carry’ state:
If the two letters are 0, M writes 0 on its third tape and moves the reading head right; if the two letters are 0 and 1, M writes 1 on its third tape and moves the reading head right; if the two letters are 1, M writes 01 on its third tape, keeps the reading head under the 1, and moves to a ‘carry’ state.
Else: (in a ‘carry’ state)
If the two letters are 0, M moves the reading head right, and transitions to a ‘no carry’ state; if the two letters are 0 and 1, M writes 01 on its third tape, and keeps the reading head under the 1; if the two letters are 1, M writes 11 on its third tape, and positions the reading head under the rightmost 1.
 - iv. M Moves the reading heads on tapes 1 and 2 one position to the left, and repeats step 1(a)iii. If at some point one of the tapes is empty and the other has letters left, M copies them to the third tape (while reversing the order and handling it correctly if in a ‘carry’ state).
 - v. When the input on the first and second tapes has been completely handled (the numbers are added), M erases the contents of tape 1, and puts the contents of tape 3 on it in a reversed order, and halt.
- (b) A 3-tape TM M that converts a binary number a on its first tape to a unary number, works as follows:
 - i. M writes 1 on its second tape, positions the reading head of the first tape under its rightmost letter.
 - ii. If the reading head of the first tape reads a 1, M copies the contents of the second tape to the third tape.
 - iii. M duplicates the string on the second tape, moves the reading head on the first tape one position to the left, and goes back to Step 1(b)ii.

iv. When the first tape is empty, M erases its first tape, copies the contents of the third tape to the first one, and halts.

2. Let M_i be a TM that semidecides L_i . M'_i that decides whether $x \in L_i$ runs all machines M_i , $1 \leq i \leq k$, “in parallel”, on string x . If M_i accepts, then M'_i accepts. If M_j , for some $j \neq i$ accepts, M'_i rejects.

Since the L_i 's form a partition, we have that $\cup_{i=1}^k L_i = \Sigma^*$. In other words, for every $x \in \Sigma^*$, $x \in L_i$ for some $1 \leq i \leq k$, in which case M_i will halt and accept x . Therefore, one of the machines M_i is guaranteed to halt and accept x , and hence M'_i always halts (and decides L_i).

3. R and RE are closed under Union, Intersection, Concatenation, and Kleene Star. To prove R languages are closed under union and intersection, run the two TM's for the two languages, and accept if either or both accept, respectively. The same technique applies to RE languages, but in the case of Union, the run of the two machines must be interleaved (for the Intersection, interleaving is not needed).

To prove R languages are closed under concatenation, the TM M (for $L = L_1L_2$) considers all possible partitions xy of string w , runs TM M_1 (of L_1) on x and TM M_2 (of L_2) on y and accepts iff both accept for some partition $w = xy$ (notice that the number of such partitions is finite). The same technique applies for RE, but M must interleave the runs of all possible partitions at the same time, to avoid getting stuck in one of the partitions, while another might accept.

To prove R languages are closed under Kleene Star, the TM M (for L^*) considers all possible partitions $w = x_1x_2 \cdots x_k$ ($0 \leq k \leq |w|$), runs M_1 (of L) on each x_i . M accepts iff M_1 accepts all x_i in one of the possible partitions. As before, a similar technique applies to RE languages, but M must interleave the runs of all possible partitions at the same time, to avoid getting stuck in one of the partitions, while another might accept.

4. The solutions to Problems (3) and (4) are combined in (3).

5. (a) L_1 is in R. TM M^* that decides the languages works as follows on input $\langle M \rangle$. It first finds the length of $\langle M \rangle$, and stores it. Then, it runs M on all inputs of length at most $|\langle M \rangle|$, for at most $|\langle M \rangle|$ steps, and accepts if M accepts at least one of the strings within the specified number of steps.

You might wonder why we limited the length of the strings. Since we bound the number of steps that M runs on an input, then there is no point on looking at any strings that are longer than that number, since if a TM is allowed to run for at most c steps, it is not possible for that TM to “process” any input symbol beyond the c^{th} symbol!

The number of possible inputs is finite, and the number of steps M runs on each input is finite, therefore M is guaranteed to halt and decide the language.

- (b) L_2 is not in RE. We prove this by a reduction from $\overline{A_{TM}}$. $\tau(\langle M, x \rangle) = \langle M' \rangle$. M' on input w : it runs M on x for $|w|$ steps; it rejects if M accepts x within $|w|$ steps, and accepts otherwise.

We now prove the validity of the reduction:

- $\langle M, x \rangle \in \overline{A_{TM}} \Rightarrow M$ does not accept $x \Rightarrow M'$ accepts all inputs, and in particular, all even numbers $\Rightarrow M' \in L_2$.
 - $\langle M, x \rangle \notin \overline{A_{TM}} \Rightarrow M$ accepts x within k steps $\Rightarrow M'$ rejects all inputs w whose length is greater than or equal to $k \Rightarrow M'$ rejects an infinite number of even numbers $\Rightarrow M' \notin L_2$.
- (c) **L_3 is not in RE.** We prove this by a reduction from $\overline{A_{TM}}$. $\tau(\langle M, x \rangle) = \langle M' \rangle$. M' on input w : it runs M on x for $|w|$ steps; it rejects if M accepts x within $|w|$ steps, and accepts otherwise.

We now prove the validity of the reduction:

- $\langle M, x \rangle \in \overline{A_{TM}} \Rightarrow M$ does not accept $x \Rightarrow M'$ accepts all strings $\Rightarrow L(M')$ is infinite $\Rightarrow M' \in L_3$.
 - $\langle M, x \rangle \notin \overline{A_{TM}} \Rightarrow M$ accepts x within k steps $\Rightarrow M'$ rejects all strings whose length is greater than or equal to $k \Rightarrow L(M')$ is finite $\Rightarrow M' \notin L_3$.
- (d) **L_4 is in R.** This is the language of all TM's, since there are no uncountable languages (over finite alphabets and finite-length strings).
- (e) • **L_5 is in RE.** M^* that semidecides the language run the two machines on ε (it interleaves the run between the machines), and accepts if both of them accept.
- **L_5 is not in R.** We prove this by a reduction from A_{TM} . $\tau(\langle M, x \rangle) = \langle M', M' \rangle$. M' on input w : it runs M on x and accepts if M accepts x .

We now prove the validity of the reduction:

- $\langle M, x \rangle \in A_{TM} \Rightarrow M$ accepts $x \Rightarrow M'$ accepts all strings, and in particular it accepts $\varepsilon \Rightarrow \varepsilon \in L(M') \cap L(M') \Rightarrow \langle M', M' \rangle \in L_5$.
 - $\langle M, x \rangle \notin A_{TM} \Rightarrow M$ does not accept $x \Rightarrow M'$ does not accept any strings, and in particular does not accept $\varepsilon \Rightarrow \varepsilon \notin L(M') \cap L(M') \Rightarrow \langle M', M' \rangle \notin L_5$.
- (f) **L_6 is not in RE.** We prove this by a reduction from $\overline{A_{TM}}$. $\tau(\langle M, x \rangle) = \langle M_1, M_2 \rangle$. M_1 is a TM that halts and accepts on any input (i.e., $L(M_1) = \Sigma^*$). M_2 on input w : it runs M on x and accepts if M accepts w .

We now prove the validity of the reduction:

- $\langle M, x \rangle \in \overline{A_{TM}} \Rightarrow M$ does not accept $x \Rightarrow M_2$ does not accept any input $\Rightarrow L(M_1) \setminus L(M_2) = \Sigma^* \Rightarrow \varepsilon \in L(M_1) \setminus L(M_2) \Rightarrow \langle M_1, M_2 \rangle \in L_6$.
- $\langle M, x \rangle \notin \overline{A_{TM}} \Rightarrow M$ accepts $x \Rightarrow M_2$ accepts all inputs $\Rightarrow L(M_1) \setminus L(M_2) = \emptyset \Rightarrow \varepsilon \notin L(M_1) \setminus L(M_2) \Rightarrow \langle M_1, M_2 \rangle \notin L_6$.

- (g) **L_7 is not RE.** Reduction from $\overline{A_{TM}}$. $\tau(\langle M, w \rangle) = \langle M_1, M_2 \rangle$.

M_2 on input x : reject. (i.e., $L(M_2) = \emptyset$).

M_1 on input y : run M on w ; if M accepts w , check whether y is of the form $\langle M', z \rangle$, where M' is a TM, and z is a string. If so, run M' on z . If M' accepts, M_1 accepts.

You can now prove that: (1) If $\langle M, w \rangle \in \overline{A_{TM}}$ then $L(M_1) = \emptyset$ (in this case $L(M_1) \leq_m L(M_2)$ since $\emptyset \leq \emptyset$, and hence $\langle M_1, M_2 \rangle \in L_7$), and (2) if $\langle M, w \rangle \notin \overline{A_{TM}}$ then $L(M_1) = A_{TM}$ (in which case $L(M_1) \not\leq_m L(M_2)$, since $A_{TM} \not\leq \emptyset$; why?).

Therefore, $\overline{A_{TM}} \leq L_7$ and hence L_7 is not in RE.

- (h) L_8 is in R. A TM M' that decides it works as follows on input x .
 If $x \neq \langle M \rangle$, where $\langle M \rangle$ is a description of a DFA, then reject.
 Construct a PDA M'' where $L(M'') = \{ww^R : w \in \{a, b\}^*\}$.
 Construct a PDA M^* where $L(M^*) = L(M'') \cap L(M)$.
 Check if $L(M^*) = \emptyset$. If so, reject; otherwise, accept. (we've seen how to decide whether the language of PDA is empty).
6. (a) **No.** L_{Σ^*} is not in SD (we saw the proof in class). Hence, L_{Σ^*} is not SD-Complete.
 (b) **Yes.** A_{TM} is in SD. We now prove that every language $L \in SD$ reduces to A_{TM} .
 If L is in SD , then there exists a TM, M that semidecides it. The reduction from L to A_{TM} is as follows:
 $\tau(w) = \langle M, w \rangle$, where M is the machine that semidecides L .
 Prove the validity of the reduction. It's simple!
7. There exists an undecidable unary language, since the number of unary languages is uncountable whereas the number of decidable languages is countably infinite! You need to know (understand and be able to prove) these facts.
8. \overline{A} is in RE, and M_1 semi-decides it.
 \overline{B} is in RE, and M_2 semi-decides it.
 We describe language C by a TM, M , that decides it:
 M on input w :
 run M_1 and M_2 on w (in interleaved mode).
 If M_1 accepts, M rejects; If M_2 accepts, M accepts.
 Notice that M always halts; we take $C = L(M)$.
9. L is in R. $L = \{0\}$ or $L = \{1\}$, and for each possibility there is a TM that decides it!
10. **NO!** Take $A = \{a^n b^n : n \geq 0\}$, and $B = a^*$, and let the reduction from A to B be: if w is of the form $a^n b^n$, erase all the b 's; otherwise, return b .
11. **May be.** Take $L_1 = \emptyset$ and $L_2 = \Sigma^*$, both of which are decidable. We know that $L_1 \subseteq A_{TM} \subseteq L_2$, and A_{TM} is not decidable. On the other hand, we know $L_1 \subseteq a^* \subseteq L_2$, and a^* is decidable.
12. (a) **NEVER TRUE.** reductions are transitive, and since A reduces to B , and B reduces to C , we conclude that A reduces to C . Therefore, if $A \notin R$, it can't be that $C \in R$.
 (b) **MAYBE TRUE.**
 (c) **CERTAIN to be TRUE.** If C is in R, and since D reduces to C , then by the reduction theorem, it follows that D is in R. Since R is closed under complement, then the complement of D is also in R.

- (d) **CERTAIN to be TRUE.** C is in RE implies that both B and D are in RE (by the reduction theorem), and so is their intersection (RE langs are closed under intersection).
13. If M is a standard single tape TM, then it is also a doubly infinite tape TM that does not move its reading head to the left of a certain cell. Therefore, M is a special case doubly infinite tape TM.

Now assume M is a doubly infinite TM, and we show an equivalent standard single tape TM M' that recognizes the same language. M' is created from M with the following modifications:

- (a) M' first puts a special symbol (that is not in the tape alphabet of M) $\#$ to the left of the input string.
- (b) After that, M' works exactly like M with the difference that whenever M' moves its reading head left and encounters $\#$, it shifts its tape contents by one cell to the right, thus vacating one cell immediately to the right of the $\#$ symbol, and places the reading head under that cell.

In other words, we simulate the doubly infinite tape TM by shifting the content of the tape to the right every time the machine has to move left of its “left boundary”.

14. A TM with *stay put instead of left* is equivalent to a finite automaton (DFA). Clearly, a DFA can be simulated by a TM of this type.

In the other direction, suppose we have a TM M of this type.

- $\delta(p, a) = (q, b, R)$ in M can be simulated by $\delta(p, a) = q$ in the NFA. Notice that remembering that the a was replaced by a b when moving right is not needed, since the TM cannot go back to the left and use that b . In other words, replacing the a by a b in M does not affect its computation.
- $\delta(p, a) = (q, b, S)$ in M can be simulated by $\delta(p, a) = q_b$ in the NFA, so that any move in M on (q, b) can be simulated in the NFA by a move on (q_b, ε) .

Therefore, M can be simulated by an NFA, and hence this type of TMs is equivalent to finite automata, and hence they recognize exactly the set of regular languages.

15. Let L be an infinite language in RE. Then, there is an infinite enumerator E for L . Let E' be an enumerator which works as follows:

- Let $x = \varepsilon$
- Run E
 - When E prints string y , if $y \geq x$ (in lexicographical order)
 - print y .
 - $x = y$.

E is an infinite enumerator. E' prints the first string that E prints. After that, E' prints strings in lexicographical order. Notice that the number of strings that are lexicographically shorter than a string x that's been already printed is finite. Since L is infinite, E would eventually print a string that is lexicographically larger or equal to x . Further, E would print an infinite number of such strings.

Therefore, E' is an infinite lexicographical enumerator of a subset L' of L . Hence, L' is an infinite recursive subset of L .

16. Let E be an enumerator for A and consider the enumerator E' which works as follows:

- Let $x = a$
- Run E
 - When E prints $\langle M \rangle$, run M on x
 - If M rejects x , print x .
 - Assign to x the lexicographically next string in Σ^* .

Let $\langle M_1 \rangle, \langle M_2 \rangle, \dots$ be the TMs enumerated by E . Notice two important properties of E' :

- (a) E' prints the i^{th} string in the lexicographical order of Σ^* if and only if M_i rejects the string. In other words, the language enumerated by E' is different from $L(M_i)$ for every M_i enumerated by E .
- (b) E' prints strings in lexicographical order.

Therefore E' is a lexicographical enumerator of a recursive language L of which none of the TMs enumerated by E is a decider.

17. We show a reduction f from A_{TM} to T . $f(\langle M, w \rangle) = \langle M' \rangle$, where M' on input x works as follows:

- If $x = ab$, accept.
- else, run M on w . If M accepts w , accept.

If M accepts w , then $L(M') = \Sigma^*$. Hence, $\langle M' \rangle \in T$.

If M does not accept w , then $L(M') = \{ab\}$. Hence, $\langle M' \rangle \notin T$.

Since $A_{TM} \notin R$, it follows that $T \notin R$.

18. The proof is in your textbook!

19. If $A \leq_m A_{TM}$ then $A \in RE$, since $A_{TM} \in RE$ (follows from the theorem we learned in class).

Now, assume that $A \in RE$. We show that $A \leq_m A_{TM}$. We need to show a function $f : \Sigma^* \rightarrow \Sigma^*$, where $w \in A$ if and only if $f(w) \in A_{TM}$. Let M be a TM that semidecides A , and let f be the function $f(w) = \langle M, w \rangle$.

If $w \in A$, then M accepts w (since M is a semidecider of A). Hence, $\langle M, w \rangle \in A_{TM}$.

If $w \notin A$, then M does not accept w . Hence, $\langle M, w \rangle \notin A_{TM}$. Clearly, f is computable. Therefore, f is a mapping reduction from A to A_{TM} .

20. Since $0^*1^* \in R$, then $A \leq_m 0^*1^*$ implies that $A \in R$.

Now, assume $A \in R$. We show that $A \leq_m 0^*1^*$. Let M be a decider for A , and let f be the function computed by the following TM M' . M' on input w : If M accepts w , then $f(w) = 01$; if M rejects w , then $f(w) = 10$. Since M is a decider, M' always halts, and f is computable.

If $w \in A$, then $f(w) = 01 \in 0^*1^*$. And if $w \notin A$, then $f(w) = 10 \notin 0^*1^*$. Therefore, f is a mapping reduction from A to 0^*1^* .

21. Let M_{3x+1} be the following TM. M_{3x+1} on input x :

While ($x \neq 1$)

$x \leftarrow f(x)$.

Let M be the TM that works as follows on its input x :

For $n = 1$ to ∞

Run H on the pair $\langle M_{3x+1}, m \rangle$.

If H rejects, halt and accept.

In other words, M is a TM that halts and accepts only if there is a number that doesn't give a sequence that ends in 1. Notice that the assumption that H exists was crucial in designing M . Otherwise, we couldn't have written "If H rejects, halt and accept."

We are not done yet. The question is: Does M halt (on any arbitrary input, since the input to M really doesn't matter)? If it does, then there is a number that doesn't give a sequence that ends in 1. If it doesn't halt, then there doesn't exist such a number; i.e., all numbers give a sequence that ends in 1. But this is easy to check. Assuming that H exists, run H on $\langle M, a \rangle$ (a is just an arbitrary input; we could have used any string). If H accepts, then the answer to the $3x + 1$ problem is NO (not all positive starting points end up a 1); if H rejects, then the answer is YES. Hence, we have a decider for the $3x + 1$ problem.

22. We reduce $\overline{A_{TM}}$ to both S and \overline{S} . To get a reduction to S , let f be the function, where $f(\langle M, w \rangle) = \langle M' \rangle$ where M' works as follows on input x :

—Obtain own description $\langle M' \rangle$ (via the Recursion Theorem)

—If $x = \langle M' \rangle$, accept

—else, run M on w .

Clearly, if M doesn't accept w , then $L(M') = \{\langle M' \rangle\}$. If M accepts w , $L(M') = \Sigma^*$. Therefore, $\langle M, w \rangle \in \overline{A_{TM}}$ iff $\langle M' \rangle \in S$. Hence, $S \notin RE$.

To get a reduction to \overline{S} , let f be the function, where $f(\langle M, w \rangle) = \langle M' \rangle$ where M' works as follows on input x :

—Obtain own description $\langle M' \rangle$ (via the Recursion Theorem)

—run M on w . If M accepts and $w = \langle M' \rangle$, accept; else, reject.

Clearly, if M doesn't accept w , then $L(M') = \emptyset$. If M accepts w , $L(M') = \{\langle M' \rangle\}$. Therefore, $\langle M, w \rangle \in \overline{A_{TM}}$ iff $\langle M' \rangle \in \overline{S}$. Hence, $\overline{S} \notin RE$.

23. (a) $C = \{L \in RE : \exists x, |x| \equiv_5 1, \text{ and } x \in L\}$. Clearly, $C \subseteq RE$. We also have that $\emptyset \in RE$ but $\emptyset \notin C$. Further, $\Sigma^* \in C$. Hence, C is a nontrivial subset of RE. By Rice's theorem, it follows that $L_C = \{\langle M \rangle : L(M) \in C\}$ is not in R. $L_C = \{\langle M \rangle : \exists x, |x| \equiv_5 1, \text{ and } x \in L(M)\} = L_1$. Therefore, $L_1 \notin R$.
- (b) $C = \{L \in RE : L \text{ contains all palindromes}\}$. You can easily prove that C is a nontrivial subset of RE, and conclude that L_2 is not in R.
- (c) $C = \{L \in RE : L \text{ does not contain any string } w \text{ such that } 001 \text{ is a prefix of } w\}$. Proving that C is a nontrivial subset of RE is trivial in this case too. It follows that L_3 is not in R.
24. (a) $S1 \rightarrow 111S, SS \rightarrow 11111$.
- (b) $111 \rightarrow \varepsilon, SS \rightarrow 1, S1S \rightarrow 11, S11S \rightarrow 111$.
- (c) $Sa \rightarrow aaS, Sb \rightarrow bbS, SS \rightarrow \varepsilon$.
- (d) $Sa \rightarrow aLA \quad Sb \rightarrow bLB$
 $aS \rightarrow aR\# \quad bS \rightarrow bR\#$
 $La \rightarrow aLA \quad Lb \rightarrow bLB$
 $X\alpha \rightarrow \alpha X \quad \text{for every } X \in \{A, B\} \text{ and every } \alpha \in \{a, b, R\}$
 $A\# \rightarrow a\# \quad B\# \rightarrow b\#$
 $LR \rightarrow \varepsilon$
 $\# \rightarrow \varepsilon$
25. (a) $S \rightarrow S_1XT$
 $S_1 \rightarrow aS_1a|bS_1b|\#TY$
 $T \rightarrow aT|bT|\varepsilon$
 $Y \rightarrow YA$
 $Aaa \rightarrow aAa$
 $Abb \rightarrow bAb$
 $Aba \rightarrow aAb$
 $Aab \rightarrow bAa$
 $AaX \rightarrow Xa$
 $AbX \rightarrow Xb$
 $YX \rightarrow \varepsilon$.

- (b) $S \rightarrow ABSC|ABS|T$
 $T \rightarrow AT|\varepsilon$
 $AB \rightarrow BA$
 $BA \rightarrow AB$
 $BC \rightarrow CB$
 $CB \rightarrow BC$
 $AC \rightarrow CA$
 $CA \rightarrow AC$
 $A \rightarrow a$
 $B \rightarrow b$
 $C \rightarrow c.$
- (c) $S \rightarrow ABSXY, S \rightarrow ABcXY$
 $YX \rightarrow XY, BA \rightarrow AB$
 $Bc \rightarrow bc, Bb \rightarrow bb$
 $Ab \rightarrow ab, Aa \rightarrow aa$
 $cX \rightarrow ca, aX \rightarrow aa$
 $aY \rightarrow ab, bY \rightarrow bb$
- (d) $S \rightarrow aBSccc|\varepsilon$
 $Ba \rightarrow aB$
 $Bc \rightarrow bbc$
 $Bb \rightarrow bbb.$
- (e) $S \rightarrow aBSd|T$
 $T \rightarrow bTc|\varepsilon$
 $Ba \rightarrow aB$
 $Bb \rightarrow bb$
 $Bd \rightarrow bd.$