CSE 34151 Theory of Computing: Homework 7
Decideability and Undecideability

Version 2: April 8, 2018

Instructions

- Unless otherwise specified, all problems from "the book" are from Version 3. When a problem in the International Edition is different from Version 3, the problem will be listed as V3:x.yy/IE:x.zz, where x.zz is the equivalent number. When Version 2 is different, it will be listed as V3:x.yy/V2:x.zz. If either IE or V2 do not have a matching number, the problem text will be duplicated.
- You can prepare your solutions however you like (handwriting, LaTeX, etc.), but you must submit them in **legible PDF**. You can scan written solutions on the printer in the back of the classroom, or using a smartphone (with a scanner app like CamScanner). It is up to you to ensure that submissions are legible. REMEMBER THAT IF WE CAN"T READ IT OR SCAN IS CUT OFF, YOU DON"T GET A GRADE FOR IT.
- The problems marked as "TEAM" may be solved in a collaborative fashion with up to 2 other students. In such cases, your submission should have the word "TEAM" at the start of the problem, followed by the names of your collaborators (must be other students in this class). When such problems are graded, the first submission encountered by the grader for the team will be used for a common grade for all identified team members.
- Please give every PDF file a unique filename.
  - If you're making a complete submission (all problems), name your PDF file `netid-hw#.pdf`, where `netid` is replaced with your `NetID` and # is the homework number.
  - If you're submitting some problems now and other problems later, name your file `netid-hw#-1-2-3.pdf`, where `1-2-3` is replaced with just the problems you are submitting now. This may be useful for team submissions.
  - If you use the same filename twice, only the most recent version will be graded.
  - The time of submission is the time the most recent file was uploaded.
- If you use LaTeX and want to draw something like a state diagram, consider using the `tikz` package. A reference document is on the website under "Assignments".
- You may also find the website http://madebyevan.com/fsm/ a useful tool for drawing state diagrams via drop and drag. It will output both .png image files and latex in the tikz format.
- Submit your PDF file in Sakai to the appropriate directory. Don't forget to click the Submit (or Resubmit) button!

Practice Problems

These problems are from the book, and most have solutions listed for them. They are listed here for you to practice on as needed and any answers you generate **should not** be submitted. You are free to discuss these with others, but you are not allowed to post solutions to any public forum.

1. V3:3.10, IE:3.17 A write-once TM
2. 3.15a decidable languages are closed under union
3. 3.16a recognizable languages are closed under union
4. 3.18 decidability and enumerators

5. 3.19 infinite decidable languages
6. 3.20 specialized TM
7. V3:4.1 Membership in decidable languages.
8. V3:4.5 Complement of E_{TM}
9. V3:4.ba,d function properties
10. V3:4.10, 4.12, 4.14, 4.25 prove language decidability

Book Exercises

These problems are found in the text book and are to be answered and submitted by each student. **If they are not marked as "TEAM," you are to solve them individually.** If they are marked as "TEAM" you may submit the same answer as the others in your team. In any case, use of solution manuals from any source or shared solutions is a violation of the ND Honor Code. You are also not allowed to show your solutions to another student not part of your TEAM.

1. (5 points) 3.4 Formal definition of an enumerator. Assume the second tape is where each string is generated before printout. Make sure you define some mechanism that signals when the string on the second tape is a "complete" string in the language, and a new string is to be started that is initially blank (like outputting a "carriage return" or using the "accept" state as a signal that second tape now includes a complete string in the language). Since there is no "input string" on the initial woek tape, for consistency let Σ be the alphabet of the strings that are printed out.
   The question "Include a definition of the enumerated language" means define what happens on a transition and exactly when the outside world knows a string is complete. How do you signal writing a character to print tape? Also ensure you discuss how the enumerator signals that for a finite language it has output the last string.
   Solution:
   Multiple answers are possible depending on how output is considered.
   Using a special print state, an enumerator is a tuple of the following components:
   - Q is set of states
   - Σ is the set of characters that can be printed out "onto the second tape"
   - Γ is the set of characters that can be written to the work tape.
   - $\delta(Q \times \Sigma \times \Gamma) \rightarrow Q \times \Sigma \times \Gamma \times \{L, R\}^2 \times \Sigma_\varepsilon$
   - q_0 is state state
   - q_{halt} is stop state
   - q_{print} is print state (akin to accept state)

   At each transition, the current state and current value under the work tape head determine the new state, what to write onto the work tape, and which way to move the work tape. It also specifies what character to write to print tape (with ε an option that writes nothing).
   Hitting the q_{print} state signals the print tape has a complete string from the language. The machine could then dump a special character to the tape, and/or clear the print tape and re position the tape head at the left.
   Another valid approach would be to add a special "carriage return" character to Σ, and then we don't need q_{print}.
2. (10 points TEAM) V3,V2:3.9; IE:3.22. Power of a K-tape PDA. Hint: consider the language $\{a^n b^n c^n\}$
3. (5 points) 3.15e decidable languages closed under intersection
   Solution:
   Let K and L be two Turing decidable languages, and let M_K and M_L denote the Turing machines deciding K and L respectively. Let $M_{K \cap L}$ denote the Turing machine deciding $K \cap L$. $M_{K \cap L}$ works as follows:
   i. On input w to $M_{K \cap L}$,
   ii. Input w to M_K.
   iii. If M_K rejects, reject.
   iv. Else Input w to M_L.
   v. If M_L accepts, accept. Else reject.

4. (5 points) 3.16b recognizable languages closed under concatenation *Solution:*
   Let K and L be two Turing recognizable languages, and let M_K and M_L denote the Turing machines that recognize K and L respectively. We construct a non-deterministic Turing machine M_{KL} that recognizes the language KL as follows:

   i. Non-deterministically cut input w into w_1 and w_2
   ii. Run M_K on w_1. If it halts and rejects, reject.
   iii. Run M_L on w_2. If it accepts, accept. If it halts and rejects, reject.

   Note the difference between the Turing machines for recognizable and decidable languages. Here, we need to take care of the fact that the machines M_K and M_L need not halt.

Non-book Problems

The following problems are not found in the text book. **If they are not marked as "TEAM," you are to solve them individually.** If they are marked as "TEAM" you may submit the same answer as the others in your team. Use of any resource you used other than the text book or class notes must be cited. You are also not allowed to show your solutions to another student.

5. (10 points TEAM) Describe how to convert any deterministic FA into a TM, i.e. how to translate each component of the tuple of a FA into a TM. In particular discuss how to translate a transition, and then rewrite the DFA of Fig. 1.12 as a TM. Show all parts of the formal definition.
   Solution:
   - The Σ for the TM is the same as from the FA.
   - Γ for the TM is the same as the Σ with the addition of a blank character $\sqcup$.
   - The set of states for the TM is the union of those from the FA plus a new accept state and new reject state.
   - The start state is the same.
   - An FA transition of $\delta(q, a) \to r$ becomes a transition in the TM of $\delta(q, a) \to (r, a, R)$ i.e. change to the same state, leave the current tape cell alone, and move to the next input cell (just like the FA).
   - For all states q in the FA's accepting set, add $\delta(q, b) \to (q_{accept}, b, L$ where b is a blank, and if we are in an accepting state in the FA and we are at the end of the input, then the TM accepts.
   - For all states q that are not in the FA's accepting state, add $\delta(q, b) \to (q_{reject}, b, L)$ where if we are not in an accepting state in the FA and we are at the end of the input, then the TM rejects.

   TM for Fig. 1.12:
   $\Sigma = \{a, b\}$
   $\Gamma = \{a, b, \sqcup\}$
   $Q_{TM} = Q_{FA} \cup \{q_{accept}, q_{reject}\}$
   $q_0 = s$
   δ:

   The conversion of Fig. 1.12 into a TM is direct. All the states stay, and each transition in Fig. 1.12 that have an "x" (where x is a or b) becomes $x \to x, R$. We add a q_{accept} state with transitions from q_1 nd q_2 of the form $\_ \to \_, L$ to q_{accept} where $\_$ is a blank. The direction on these last transitions is irrelevant.
6. (15 points TEAM) SAT. The class on SAT described how to convert Sudoku puzzles into SAT problems, Both 9x9 and 2x2 examples were given. For this problem, consider a 3x3 problem where each of the 3 rows, 3 columns, and 2 diagonals must contain some ordering of 1, 2, and 3.
   (a) (8pt) Describe at the level of detail in the notes how to construct a corresponding general 3x3 SAT expression. How many variables? What are the different classes of clauses that are present (and for each class, how many are there - you need not write out each clause in detail but describe

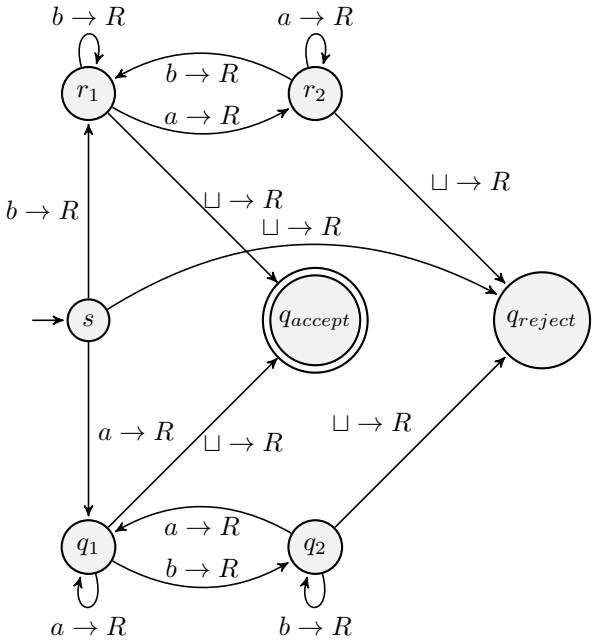

Figure 1: The TM for FA in Fig. 1.12

a general approach for generating them).

(b) (5pt) Assume initial conditions where there is a "1" in the upper left and a "3" in the lower right. Discuss how you might, by hand without an explicit algorithm, deduce what the solution of the SAT problem is. For example, is there is some clause where all but one literals are already known to be false, then you can force the remaining clause to be true. For this problem, you can in fact get to a (in fact the only) solution this way.

(c) (2pt) At a very high level how would you write an algorithm which would find some solution if only the "1" in the upper left corner is initially specified.

Solution: Solutions:

Define $3x3x3 = 27$ variables $x_{i,j,d}$ $1 \leq i, j, dl3eq$ where i and j are indices into the 3x3x grid and d is a digit.

Following are the sets of clauses:

(a) 9 clauses: 1 for each cell (i,j) to ensure it has a digit: $(x_{i,j,1} \vee x_{i,j,2} \vee x_{i,j,3})$

(b) 9 sets of 3 clauses: for each of $1 \leq d < d' \leq 3$, $(\neg x_{i,j,d} \vee \neg x_{i,j,d'})$

(c) To state that row i, for example, has all 3 digits

- 3 clauses (1 for each value of d to ensure its somewhere in row) where d'th clause is $(x_{i,1,d} \vee x_{i,2,d} \vee x_{i,1,d})$
- 3 sets of 3 clauses to ensure uniqueness $1 \leq j < j' \leq 3$, $(\neg x_{i,j,d} \vee \neg x_{i,j',d})$

(d) Repeat above for each of 3 rows, 3 columns, and 2 diagonals (8 sets). Since each set is 3+3x3 = 12 clauses, we need $8x12 = 96$ clauses

Total of $9 + 9x3 + 96 = 132$ clauses.

(b) Initial conditions add $(x_{1,1,1})$ and $x_{3,3,3})$, which requires both variables to be true. This will force $x_{2,2,2}$ to be true (from the diagonal clauses). Then, for example, to find values for column 1, row 2 requires that either $x_{2,1,1})$ or $x_{2,1,3})$ is true (but not both). Likewise row demands that either $x_{3,1,1})$ or $x_{3,1,2})$ is true (but not both). But $x_{3,1,1})$ cannot be true, thus $x_{3,1,2})$ must be true, which forces $x_{2,1,3})$ to be true. Same reasoning drives other cells.

(c) Will have to pick a cell and try assigning a value, and see if that leads to a solution as above. If not we must try a different value for that cell.