In-Class Semi-Workshop Problems for 2018/4/2, plus a Template and a Worked Example of Using the Template

(last updated 2018/3/25/906pm)

CSC 280: Computer Models & Limitations Spring 2018
Instructor: Lane A. Hemaspaandra Grad TA: Andrew Read-McFarland

WARNING: These problems generally require things you will have learned in the 3/28 class, which corresponds to Chapter 5.1 of SIP. So you will probably want to wait until right after the 3/28 class to start on these. However, if you have a 3/28 workshop, you will want to before that workshop—even if is the one right after class—at least be aware of, and bring into the workshop a printout of, the template provided here, as you’ll be using that template extensively at the 3/28–3/29 workshops, and on your exercise set 9.

1 Problems (Most Are Challenging At Least Until You Get the Hang of Things)

Note: You should read Section 2 of this handout, the Hint/Framework/Example section, before doing these problems.

I urge everyone, and all groups, to try all the problems. But so that at least one group is “on point” on each, I’ve beside each problem put the name of one group as the on-point group for that problem. Please either solve your group’s problem at your 3/28–3/29 workshop, or if you don’t do it then, then between then and 4/2, and have some member of your group come in with a solution, on paper, to project on the document cam, at the 4/2 class.

General directions: For each language below, show it is undecidable. Please make sure to use the Sipser Chapter 5.1 flavor of proof. So assume it is decidable. And then use a hypothetical total TM (decider) for it to let you show that $A_{TM}$ is decidable, and note that that is a contradiction.

1. [W-early-B] $L_1 = \{ \langle M_1, M_2, M_3, M_4 \rangle \mid$ each of the $M_i$’s is a TM and $L(M_1)$ is finite and $L(M_2)$ is infinite and $L(M_3) = \emptyset$ and $L(M_4) = \{ \varepsilon \} \}.$

2. [W-early-S] $L_2 = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is finite and (or cand) } ||L(M)|| = 280 \}. \text{ Note: } ||S|| \text{ denotes the number of elements in } S.$

3. [W-late] $L_3 = \{ \langle M \rangle \mid M \text{ is a decider (i.e., is a total TM) } \}.$ Please see the note attached to the next problem, to help you understand what this problem is about.

4. [T-early] $L_4 = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is decidable } \}.$ Note that this is not the same problem as the previous one. This is about $L(M)$ being decidable, and the previous one was about $M$ being a decider.
5. [T-late-B and T-late-S (the two groups can, after both solve it, decide between themselves which gets the adventure of putting up a solution on the document cam, but both groups should, separately from each other, solve it)] \( L_5 = \{ (M, x) \mid M \text{ is a TM and } M \text{ contains the state } q_7 \text{ and on input } x \text{ the TM } M \text{ enters the state } q_7 \text{ at least 2018 times } \}. \)

2 Hint/Framework/Example

If the above problems for the 4/2 class are going well and you’re already writing perfect proofs of this sort, then great and you in that case perhaps don’t need this hint/framework/example. (Nonetheless, using the framework is very helpful, and indeed on Exercise Set 9 and perhaps elsewhere, I may outright require its use.)

However, if you are having trouble with them, or just want a framework in which to (usually) do problems, I’ll include in a moment a bit of “framework” text (plus a pointer to a worked example of using that framework). What I mean by that is you may well want to fit all your answers to all the problems for the above problems into the pattern that that framework text sets out.

I’m not guaranteeing that this text will solve every problem in the universe. However, it is sort of the canonical framing of the “contradiction” approach, and in my own jotted-down notes on each of the 4/2 problems, I have used basically this framing and it worked fine.

So... here is the framework text. The ??? and ???[comment]??? marks are places where you’ll need to fill in text! (You can probably copy-paste this to form the start of your answers, though do be careful that your copy-paste does not garble letters or math symbols. But it is basically plain-text, on purpose, to try to make it easier to copy.)

THE TEMPLATE:

[We’re trying to prove some particular set, \( G \), is undecidable.]

Our decider for \( A_{TM} \) assuming that the above set \( G \) is decidable by machine \( R \) is this machine/program \( S \):

\( S: \)

On input \( <M, w> \) build the machine \( M' \) that itself, when its input is \( y \),

???[here you have to describe the program of \( M' \), that is, what it does on input \( y \)]???

Simulate \( R(???) \)[the argument to \( R \) had better match the syntax/type of the problem we’re attacking!]???, and ???[here you say what we do, e.g., "if \( R \) accepts we accept and if \( R \) rejects we reject" or "if \( R \) accepts we reject and if \( R \) rejects we accept" are the most typical things one would have here]???

End of the program \( S \)

???[here goes your explanation of why \( S \) is a decider for \( A_{TM} \). do NOT forget this part of our template. it should have 3 subparts:]

(1) argue that \( S \) is a decider.
(2) argue that if \( <M, w> \)\( \in A_{TM} \) then \( S \) accepts on input \( <M, w> \).
(3) argue that if \( <M, w> \)\( \not\in A_{TM} \) then \( S \) rejects on input \( <M, w> \).
NOTE: (2) and (3) actually basically give (1), give or take the issue of syntactically illegal inputs to $M'$. But still, just as an automatic thing, do include (1), (2), and (3).

WARNING: The above template assumes one just calls $R$ on one input. There exist examples where one would have to call $R$ lots, and then do more complex things as to how to turn the outcomes from those runs into an accept/reject action. But the above simplified template covers the vast majority of what one would ever face (and of course, even if I tell you to use the template, if the use of it requires an adjustment of the sort I just mentioned, you will need to make such an adjustment).

We will soon do the promised example.

But first a comment. At times (perhaps on exercise sets or on Midterm II), you may be asked some pretty hard questions. For example, when building the machine $M'$ of the template above, for some of the problems you might NOT be able to assume that $M'$ ignores its input! Other problems might just be quite different in look and flavor from the problems we’ve typically been doing. You’ll need to really understand how to do these problems—not just to have memorized some examples. Please study and practice (and use office hours) well, until you’re pretty comfortably on top of this material.

And on we go to the promised example. That is, I will now give a quick example of a problem where $M'$ cannot ignore its input, and will using the template prove that that problem is not decidable.

So, recall that our template is as stated above.

Question: Prove that $G = \{ <M> | L(M) \text{ is NOT a context-free language} \}$ is undecidable.

Answer: Our decider for A_TM assuming that the above set $G$ is decidable by machine $R$ is this machine/program $S$:

$S$:

- On input $<M,w>$ build the machine $M'$ that itself, if its input is $y$, simulates $M$ on input $w$ and if that rejects we reject, and if that runs forever we obviously run forever as we simulate it, but, crucially, if that accepts, instead of ourselves accepting, we instead accept if $y$ is of the form $zz$ for some string $z$ and otherwise we reject.

$\text{Simulate } R(<M'>)$, and if $R$ accepts we accept and if $R$ rejects we reject.

End of the program $S$

$S$ is clearly a decider, basically because $R$ is a decider and $S$ just builds a machine in a doable way and then simulates the decider $R$ on that and then accepts/rejects in a simple way based on the action of $R$.

If $<M,w>$ is in A_TM, then $L(M')$ is $\{zz \mid z \in \Sigma^\star\}$, which
is NOT a CFL, and so R will accept [sic] on input <M’>, and so S will accept on input <M,w>.

If <M,w> \not\in A_{TM}, then L(M’) = \emptyset, which is not NOT [sic!!!] a CFL (i.e., it is a CFL), and so R will reject [sic] on input <M’>, and so S will reject.