CSE491/596, Fall 2022 **Problem Set 3** **Due Thu. Oct. 20**

Lectures and Reading.

Dick Lipton and I have sometimes advanced the view that *reductions* between problems should be regarded as fundamental objects of study, even more than the problems themselves. We mean this more at the level of computational complexity theory. But for the level of (un-)decidability, too, it is important to understand the basic mechanism of a reduction. This goes especially for understanding the reduction from A_{TM} to NE_{TM}m whose "All-Or-Nothing Switch" property can be re-used to carry out numerous other reductions. I have decided again to recommend the notes

https://cs.uwaterloo.ca/ watrous/ToC-notes/

by John Watrous, whose chapters 15,16,17 are individually linked from the course webpage. My overall main point is that reductions $A \leq_m B$ have three levels of reasoning:

(a) There is the "outer level" where the reasoning goes: "If there were a Turing machine S that decides B, then the reduction would give us a total Turing machine R that decides A. R would have the form, 'On input x, run the reduction function to get $y = f(x)$ and run S on y. If S accepts, accept. If S rejects, reject.' But we know A is undecidable, so R cannot exist. Thus S cannot exist. So B is undecidable too." *Well, please do not ever write this.* The "Reduction Theorem" at the end of the Monday 9/30 lecture takes care of this reasoning once-and-for-all.

(b) The "middle level" is the logical reasoning that establishes the equivalence $x \in A \iff f(x) \in B$, often best pictured as two implications $x \in A \implies f(x) \in B$ and $x \notin A \implies f(x) \notin B$.

(c) The "inner level" is the code editing used to design f which makes the implications come true.

Most sources I find online strike me as blending (a) and (b) together in a way that detracts from (c), but Watrous's notes not only treat (b) and (c) cleanly but also match other aspects of my lectures well. Hence please also read chapters 15–17 of his notes, focusing on ch. 16. Here are some things to watch for:

- In ch. 15, his A_{DTM} is the same as A_{TM} since a TM defaults to being deterministic. The decidable language S_{DTM} which he gives in equation (15.2) will figure into the two-tape DFA lecture mentioned above, and later into the equivalence of two definitions of the class NP. (It is a variant of the historical predicate and function combo $T(M, w, \vec{c}) \wedge U(\vec{c}) = y$—meaning that $\vec{c}$ is a string encoding a valid halting (or accepting) computation of the TM M on input w and its output is y—called the *Kleene T-predicate* after the same Stephen Kleene who proved the theorems on regular expressions and FAs covered earlier.)

- Note that Watrous likewise uses the diagonal language as his first example of undecidability, calling it DIAG rather than D_{TM} or just D. He does not define K_{TM} as the complement of D_{TM} but shows the undecidability of A_{TM} directly, using "K" to refer to a transformed machine which makes an appearance in ch. 16 too as the edited code for the mapping reduction from A_{TM} to HP_{TM} (which he calls HALT).

- Section 15.4 has Theorem 15.7—covered in my lectures as RE $\cap$ co-RE = REC—and the trick of "hard-coding an input string." I used that trick earlier to convert the "Universal RAM Simulator" $U(P, x) = P(x)$ into a Turing machine M_P which simulates a particular program P by way of writing P on its own tape using hard-coded states and then going to the start state of U, so as to make $M_P(x) = U(P, x)$.

- Note that Watrous devotes time in his Chapter 16, Example 16.6, to the $A_{TM} \leq_m HP_{TM}$ reduction, which was covered at the end of the Friday Oct. 4 lecture. His example 16.7 follows from that lecture upon realizing the equivalence to the mirror-image reduction of their complements: $K_{TM} \leq_m NE_{TM}$. His example 16.8 is an example of re-using the "all-or-nothing switch" reduction. His example 16.9 will be presented this week as a way to build on the "switch" mechanisms.

- The material in sections 17.1–17.3 has mostly been touched on, while the last section 17.4 will be covered in-tandem with Theorem 10.2 from Arun Debray's notes and the analogous theorem for NP, which is Theorem 13.12 beginning the last section of Debray's notes which was also given out last Friday.

So the reading for now in brief is • the newly-given Watrous chapters 15–17, and • the corresponding coverage in sections 8 and 9 of Debray's notes (but it is rather telegraphic). Skim the Post Correspondence Problem for now and jump ahead to Theorem 10.11 ("Rice's Theorem") and its proof. Then skip the Recursion Theorem and sections 11–12; part of the latter is covered in a different way in an extra-credit problem on this assignment. After the coverage of reductions finishes on the 14th or 15th, we will start on complexity theory using section 13 of Debray's notes and parallel notes by myself and Eric Allender and Michael Loui which will be given out in hardcopy. That material will not be on the first exam—only through this homework and section 10 of Debray's notes.

Assignment 3, due Thu. 10/20 in class

(1) Define the *reversal* of a language A by $A^R = \{x^R : x \in A\}$, where x^R is the reversal of the string x. Give a decision procedure for the following computational problem:

INSTANCE: A DFA M.

QUESTION: Does $L(M)$ equal its reversal?

Note that this would be true if every string in $L(M)$ were a palindrome, but it is true if $L(M) = \Sigma^*$ too. You should begin by morphing the DFA M into an NFA N such that $L(N) = L(M)^R$ by making each arc go the other way, making $F_N = \{s_M\}$, and making N have a new start state with ϵ-arcs to all the old final states of M. Ultimately your algorithm should produce a DFA M' such that deciding the E_{DFA} problem on M' tells you the answer about M. (18 pts.)

(2) The feature of allowing a Turing machine to keep its head stationary (S for "stay" in an instruction) is a useful convenience but not necessary: An instruction (p, c, d, S, q) could always be replaced by making a special state p' and giving the instructions $(p, c, d, R, p'), (p', -, -, L, q)$ instead, where the '$-$' is a "wildcard"—or rather, is shorthand for actually having an instruction (p', a, a, L, q) for every $a \in \Gamma$. Executing an instruction with "stay" is hence an elective rather than essential capability of programs. Now consider the following languages:

(a) $L_a = \{\langle M, w\rangle : M(w)$ executes a stay instruction$\}$.

(b) $L_b = \{\langle M\rangle :$ for some string w, $M(w)$ executes a stay instruction$\}$.

(c) $L_c = \{\langle M\rangle :$ for all strings w, $M(w)$ executes a stay instruction$\}$.

(d) $L_d = \{\langle M\rangle :$ for all strings w, $M(w)$ never executes a stay instruction$\}$.

(e) $L_e = \{\langle M \rangle :$ for all strings w, $M(w)$ can never execute a stay instruction, for the simple reason that it has no stay instructions in its code$\}$.

Classify each language according to whether it is (i) decidable, (ii) c.e. but not decidable, (iii) co-c.e. but not c.e., or (iv) neither c.e. nor co-c.e. Once you give a reduction for the first one or two undecidable cases, you should be able to handle the others with minimal further effort, making use of a close analogy with the A_{TM} and/or E_{TM} problems which your first reduction(s) should establish. (Note: The answer to L_d can be "yes" even if the computation $M(w)$ never halts, so long as it never executes an instruction with S, 6+9+6+9+6 = 36 pts.)

(3) Consider the following decision problem:

INSTANCE: A deterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, \Box, s, q_{acc}, q_{rej})$.

QUESTION: Is there an input x that causes M to visit every one of its states at least once before halting?

Prove by reduction from A_{TM} that this problem is undecidable. (18 pts., for 72 on the set. *Hint:* You are allowed to make Γ bigger in your M' than it is in the given M. But for 6 pts. extra credit, explain how you could adapt the reduction in case all TMs were limited to the characters 0, 1, and the blank—ideas from problem (2) might come in handy.)