Homework 8: Undecidable languages

CSE 30151 Spring 2016

Due 2016/04/12

Instructions

Please note that you will **lose one point** if you don't follow these instructions.

- You can prepare your solutions however you like, but you must submit them as a single PDF file.
- Please name your PDF `netid-hw8.pdf`, where `netid` is replaced with your NetID, or `netid-hw8-1234.pdf`, where `1234` is replaced with the problems you are submitting.
- If you use the same name twice, only the most recent version will be graded!
- Submit your PDF file in Sakai. Don't forget to click the Submit (or Resubmit) button!

Problems

1. ***The Power of 10.*** Look at *The Power of 10* (`http://bit.ly/powof10`), a set of rules for writing mission-critical code developed at JPL. Use a diagonalization argument to show that there exists a decidable language L that cannot be decided by a program that complies with these rules. Note: Please ignore the exception that reads: "This rule does not, of course, apply to iterations that are meant to be nonterminating—for example, in a process scheduler. In those special cases, the reverse rule is applied: It should be possible for a checking tool to prove statically that the iteration cannot terminate." You need to design L and write your argument in three parts:

   (a) Describe L by writing a program (in pseudocode) that decides it. Assume that your program includes the following two functions, which you don't have to write:

   - `check(m)`: returns true if the string `m` is the source code of a program that is syntactically correct and complies with *The Power of 10*; otherwise, returns false.

   - `run(m, w)`: runs the program whose source code is the string `m` on the input string `w`, and returns true if `m` accepts `w`; otherwise, returns false.

   (b) Explain why your program always halts.

   (c) Show that there does not exist a program that complies with *The Power of 10* and decides the same language that your program does.

2. **Bounds checking.** [Problem 5.14] Show that it is undecidable whether a Turing machine M, on input w, ever attempts to move its head past the left end of the tape. Your answer should be a reduction from another undecidable problem (don't use Rice's Theorem).

3. **More bounds checking.** Show that it is decidable whether a Turing machine M, on input w, ever attempts to move its head past the right end of the input string w. Your answer should be a construction of a TM – a high-level description is enough.

4. **Rice's Theorem.** Let P be any nontrivial property of Turing-recognizable languages: that is, P is a subclass of the class of Turing-recognizable languages that is neither empty nor equal to the class of all Turing-recognizable languages.

   Rice's theorem [Problem 5.28] says that it is undecidable, given a Turing machine M, whether the language M recognizes has property P.

   Once you understand the statement of Rice's theorem, then the following problems should be easy (don't overthink them):

   (a) [Problem 5.30c] Use Rice's Theorem to prove that it is undecidable whether a Turing machine M accepts the language Σ^*.

   (b) [Problem 5.29] Show that both conditions in Rice's Theorem are necessary, by:

   - showing that the two trivial properties are decidable;
   - giving an example of a property of Turing machines – as opposed to the languages they recognize – that is decidable.