CS3102 Theory of Computation
Problem Set 6
Department of Computer Science, University of Virginia

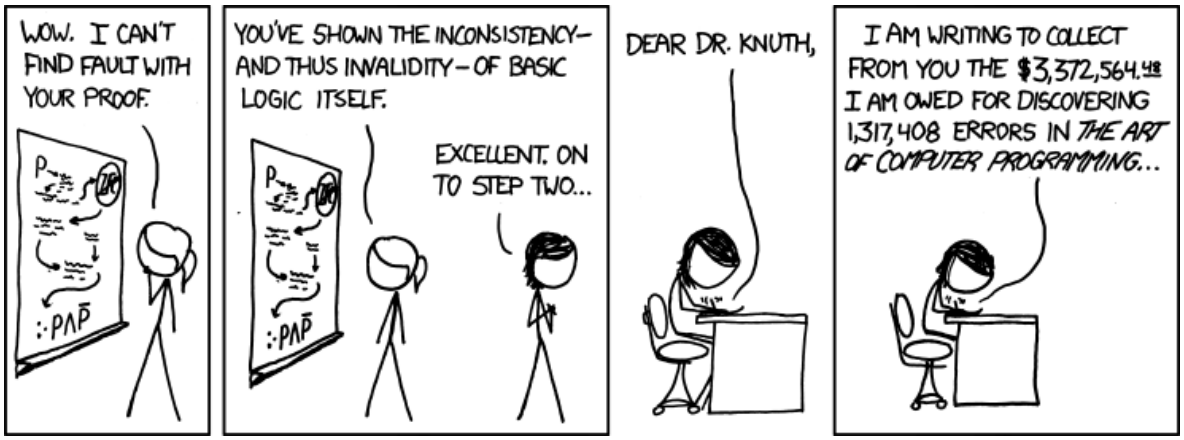
Gabriel Robins

Please start solving these problems immediately, don't procrastinate, and work in study groups. Please prove all your answers; informal arguments are acceptable, but please make them precise / detailed / convincing enough so that they can be easily made rigorous if necessary. To review notation and definitions, please read the "Basic Concepts" summary posted on the class Web site, and also read the corresponding chapters from the Sipser textbook and Polya's "How to Solve It".

Please do not simply copy answers that you do not fully understand; on homeworks and on exams we reserve the right to ask you to explain any of your answers verbally in person (and we have exercised this option in the past). Please familiarize yourself with the UVa Honor Code as well as with the course Cheating Policy summarized on page 3 of the Course Syllabus. To fully understand and master the material of this course typically requires an average effort of at least six to ten hours per week, as well as regular meetings with the TAs and attendance of the weekly problem-solving sessions.

This is not a "due homework", but rather a "pool of problems" meant to calibrate the scope and depth of the knowledge / skills in CS theory that you (eventually) need to have for the course exams, becoming a better problem-solver, be able to think more abstractly, and growing into a more effective computer scientist. You don't necessarily have to completely solve every last question in this problem set (although it would be great if you did!). Rather, please solve as many of these problems as you can, and use this problem set as a resource to improve your problem-solving skills, hone your abstract thinking, and to find out what topics you need to further focus on and learn more deeply. Recall that most of the midterm and final exam questions in this course will come from these problem sets, so your best strategy of studying for the exams in this course is to solve (including in study groups) as many of these problems as possible, and the sooner the better!

Advice: Please try to solve the easier problems first (where the meta-problem here is to figure out which are the easier ones ☺). Don't spend too long on any single problem without also attempting (in parallel) to solve other problems as well. This way, solutions to the easier problems (at least easier for you) will reveal themselves much sooner (think about this as a "hedging strategy" or "dovetailing strategy").



1. The following problems are from [Sipser, Second Edition]:

   Pages 159-162: 3.4, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.17, 3.18, 3.19, 3.22

   Pages 182-184: 4.2, 4.3, 4.4, 4.6, 4.7, 4.9, 4.12, 4.15, 4.18, 4.19, 4.22, 4.26, 4.27, 4.28

2. Construct a context-sensitive grammar that generate $\{a^n b^n c^n \mid 1 \leq n\}$. Make your grammar as "small" as possible in terms of the number of non-terminals and productions in it.

3. Construct a context-sensitive grammar that generate $\{a^{(n^{\wedge}n)} \mid 1 \leq n\}$. Make your grammar as "small" as possible in terms of the number of non-terminals and productions in it.

4. Give (and prove) several example non-Turing-recognizable languages.

5. Describe a Turing machine that prints out its own description (regardless of its input).

6. Let $L=\{0^k \mid k \text{ is a Fibonacci number}\}$. Describe a Turing machine that accepts L. Give a (context-sensitive) grammar that generates L.

7. Describe a two-tape Turing machine that prints out on its second tape only prime numbers (in either binary or unary, separated by commas), such that every prime number will eventually be printed there.

8. Describe a two-tape Turing machine that prints out on its second tape valid encodings of all Turing machines (separated by commas), such that every Turing machine (including itself) will eventually be printed there.

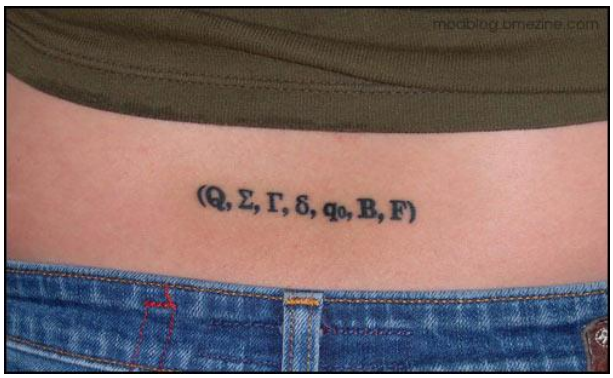
9. Modify Turing machines so that they can insert new tape cells into their tape(s), and also remove old tape cells from their tape(s), instead of only (over)writing existing tape cells. (a) Define carefully the transition function and the computational behavior of such machines. (b) Show that such a machine can be simulated by an ordinary Turing machine with at most a quadratic loss of efficiency.

10. True or false: any two-tape Turing machine that uses constant space (aside from the read-only space occupied by the input string) recognizes a regular language.

11. True or false: if L is Turing recognizable, then there is a Turing machine M that enumerates L without ever repeating an element of L.

12. Is the set of non-finitely-describable real numbers closed under addition? Squaring?

THE BIG PICTURE SOLUTION TO THE HALTING PROBLEM



13. Are the decidable languages closed concatenation? Union? Complementation? Kleene closure?

14. Are the recognizable languages closed concatenation? Union? Complementation? Kleene closure?

15. Are the non-recognizable languages closed concatenation? Union? Complementation? Kleene closure?

16. Are the non-finitely-describable languages closed concatenation? Union? Complementation? Kleene closure?

17. Is a countably-infinite union of decidable languages necessarily decidable?
Is a countably-infinite intersection of decidable languages necessarily decidable?

18. Is a countably-infinite union of recognizable languages necessarily recognizable?
Is a countably-infinite intersection of recognizable languages necessarily recognizable?

19. Is a countably-infinite union of non-finitely-describable languages necessarily non-finitely-describable?
Is a countably-infinite intersection of non-finitely-describable languages necessarily non-finitely-describable?

20. What is the infinite union of all the decidable languages?
What is the infinite intersection of all the decidable languages?

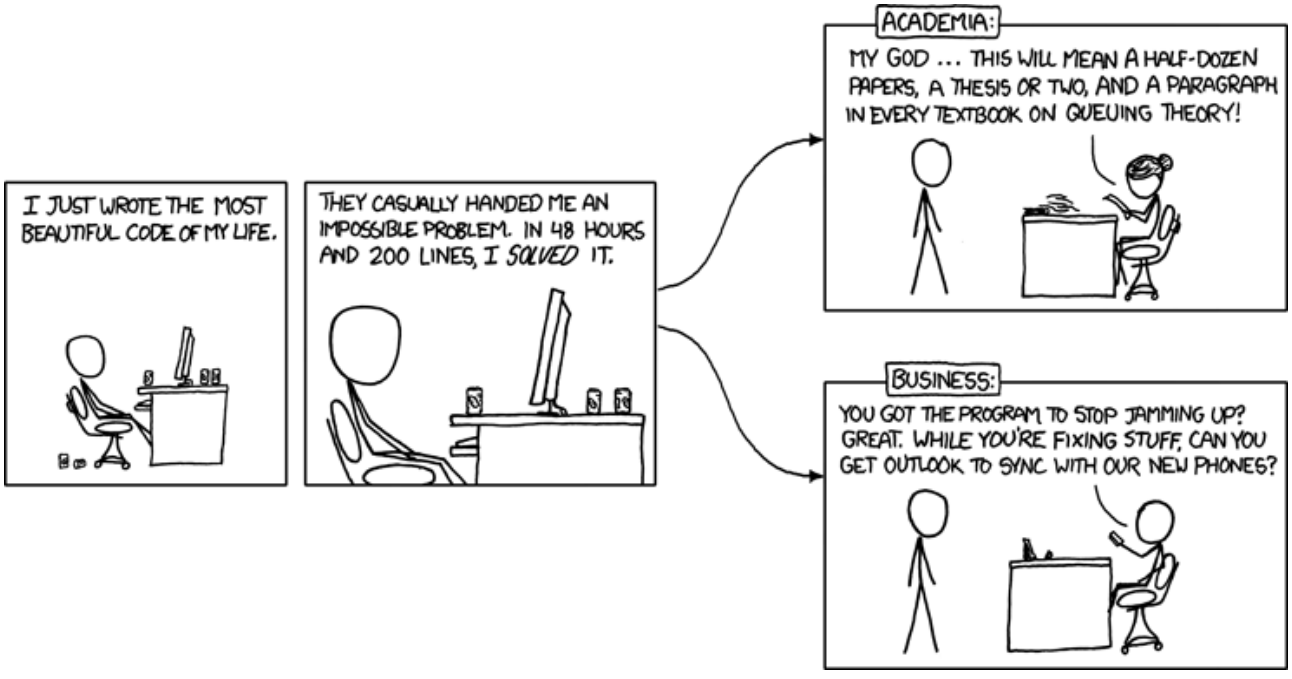
21. What is the infinite union of all the recognizable languages?
What is the infinite intersection of all the recognizable languages?

22. What is the infinite union of all the non-finitely-describable languages?
What is the infinite intersection of all the non-finitely-describable languages?

23. Can a non-computable number be rational? Must an irrational number be non-computable?

24. Let YESNO(L)={xy | x∈L and y∉L, x,y∈Σ^*}. Does YESNO preserve decidability?

25. Let PALI(L)={w | w∈L and w^R∈L}. Does PALI preserve decidability?



26. Let F denote some finite language, R denote some regular language, C denote some context-free language, D denote some decidable language, E denote some recognizable language, and N denote some non-recognizable language. For each one of the following statements, prove whether it is <u>always</u> true, <u>sometimes</u> true, or <u>never</u> true:

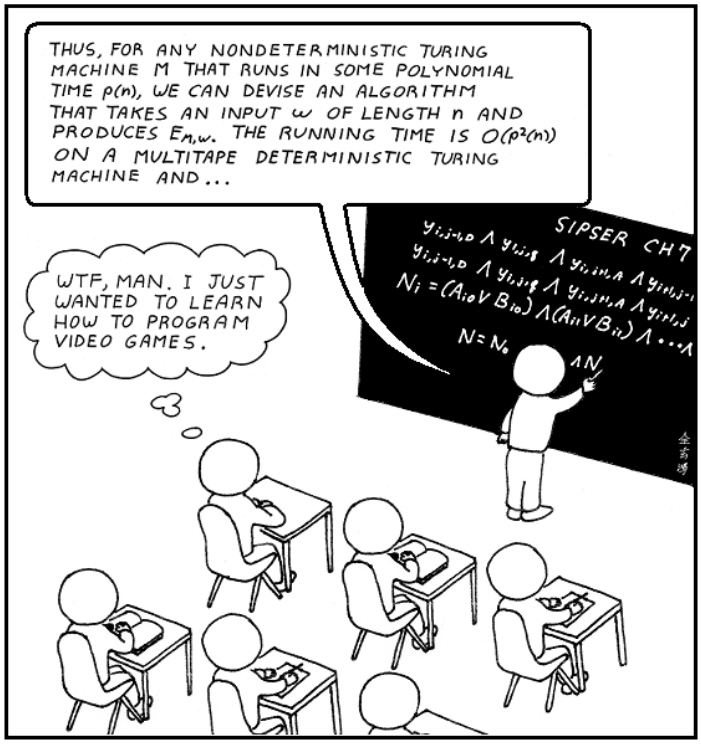
a) $(D \cup R)^*$ is decidable
b) RD is regular
c) N – F is decidable
d) $N^* \cap E \cap C \cap F$ is regular
e) N^* is context-free
f) SHUFFLE(D,C) is decidable (where SHUFFLE is defined in problem 19 on Set 5).
g) E is NP-complete

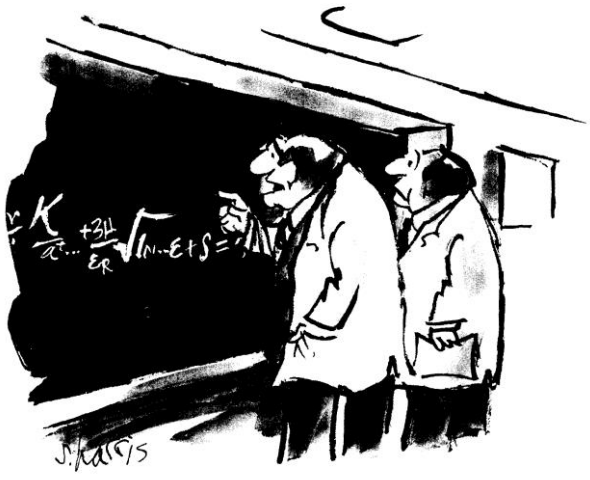
27. Define a deterministic "**infinite automata**" similarly to a deterministic finite automata, but where the state set Q is no longer restricted to be finite (i.e. can be countably infinite), and all other aspects of the infinite automata remain similar to their finite counterparts.

(a) Characterize precisely the class of languages accepted by deterministic infinite automata.
(b) Characterize precisely the class of languages accepted by non-deterministic infinite automata.
(c) Do oracles increase the power of infinite automata?

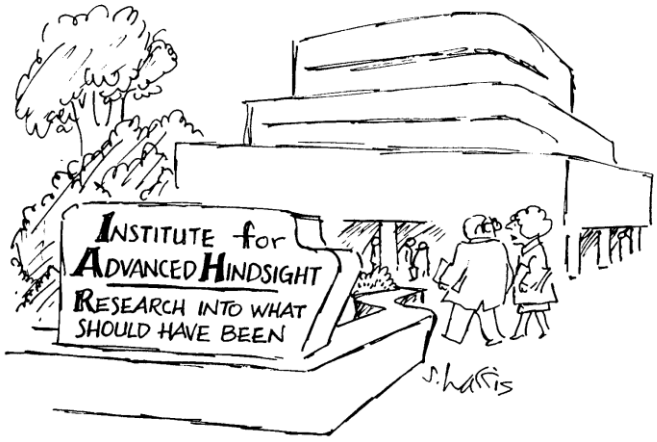
28. Let $L=\{0^k \mid k \text{ is a Fibonacci number}\}$.

(a) Describe a Turing machine that accepts L.
(b) Give a (context-sensitive) grammar that generates L.





"Very creative. Very imaginative. Logic…*that's* what's missing."



" 'Look', I would say to Leonardo. 'See how far our technology has taken us.' Leonardo would answer, 'You must explain to me how everything works.' At that point, my fantasy ends."