

Name: \_\_\_\_\_

## Exercise B6/C0, 3/1/05

This exercise is due on Tue., Mar. 15, 2005 (i.e., after the spring break).

### The Computational Power of Turing Machines

We have been using the Turing machine as our model of computation. It is convenient because with TMs, we can capture all of (and exactly) the algorithms as we informally understand (Church-Turing thesis). We also discussed that nontrivial properties of TMs are undecidable, i.e., not captured by the TM (Rice's Theorem). In addition to the main problems, we noted more problems in this category in Module B Review Exercise. So, we know that TMs are limited, esp. when we need to deal with, say, practical questions involving "big" issues in the CS industry. On the other hand, we can easily imagine situations where even the TM seems to be overkill. For example, to check whether or not a given string is an acceptable variable name (e.g., in a compiler), we do not need the full power of the TM (i.e., no need for the infinite tape). If we can solve a problem with an easier means, we should use it instead of unnecessarily complicated means ("principle of parsimony" and "Ockham's razor"). The ability to identify and use the minimal mechanism for a given problem is essential in virtually any area of study or work. Do you have this skill? This is another property that would distinguish computer scientists from programmers. "Elegant" solutions are often sought not because of esthetics but because of practicality, e.g., for easy understanding and maintenance.

**Task:** Write a concise essay about whether or not the TM is an appropriate model *with respect to its computational power*, referring to (i) your problem from Exercise 00 and (ii) your mini research question(s) from Exercise A6/B0 or different one(s) from the list (below). In addition to discussing these problems in their entirety, you must also consider possible subproblems (cf. **Note 1** below). Apply the skills learned in earlier CS courses, e.g., object orientation and modular programming, and analyze whether any of the problems/subproblems would need the full power of the TM.

**Note 1:** We know that compilation is a decidable problem. Thus, a TM is surely sufficient for the problem. But as we briefly discussed in class, a compiler typically consists of multiple phases/modules. When these phases are considered as subproblems, they would require mechanisms of different complexity. For example, the lexer requires a DFA (but not a full TM or even a stack), the parser requires a stack (but not a full TM), and semantic analysis (e.g., variable reference) requires some sort of table (may correspond to a TM). This way, when we deal with a simpler problem, e.g., lexical analysis, we can do it as easily/fast as possible. In a similar manner, your problem(s) may be broken into subproblems, which may require different mechanisms of different complexity.

**Note 2:** For each of the research questions listed below, there is some connection to the Theory of Computation. Although algorithmic computation is limited as we observed in Modules A & B, being able to identify such limitation could let us pinpoint the source of complexity beyond the traditional computation. With this ability, one will be able to learn a broad range of subjects in a systematic manner. When I first studied the Theory of Computation on my own almost two

decades ago (no CS courses taken and no plan to study CS at that point), the motivation was to understand the mechanism behind human language (i.e., linguistics). But the Theory certainly changed the way I understand human language (well, at least certain systematic aspects of it). When I started my graduate work in CS and studied CS topics such as programming languages, compilers, etc., I felt “OK.” But when I studied cognitive science and complex systems, it was the limitations of the Theory of Computation that I appreciated. It may be difficult to “use” the Theory in this manner, esp. during this semester (we do not really discuss much, outside the Theory topics in this course). But I hope you keep this in mind and try to apply the acquired ideas throughout your career. Whether you can connect apparently different things and analyze them systematically would make a huge difference in your life, regardless of your career choice.

### List of sample research questions from Exercise A6/B0

1. Can **organizational dynamics** be modeled as an algorithm?
2. Can **evolution** be modeled as an algorithm?
3. Can **ecology** be modeled as an algorithm?
4. Can **human development** be modeled as a computer?
5. Can our **minds** be modeled as a computer?
6. Can **vision** be modeled as an algorithm?
7. Can **learning** be modeled as an algorithm?
8. What would be the minimal mechanism to process **human language**?
9. Can the entire situation of an arbitrary **game** be modeled as an algorithm?
10. Would “perfect” **user modeling**, e.g., for web search, be possible?
11. Would “perfect” **computer security** be possible?
12. Can the entire process of **software engineering** be modeled computationally?
13. Can **computer networks** be modeled as a single computer?
14. Could **biology** be reduced to physics?
15. Could some computer generate real (not pseudo) **random numbers**?
16. Would it be possible to decide whether the given numbers are **random**?
17. Would **randomization** affect computability and/or complexity?
18. Would **parallelism** affect computability and/or complexity?
19. Would **artificial neural network** be more powerful than TMs?
20. Would **cellular automata** be more powerful than TMs?
21. Would the use of **analog** (or fuzzy) values affect computability?
22. Would relativistic, quantum, or some other **modern-physics**-based computation surpass TMs?
23. Can all the cases of **on-line algorithms** be simulated by off-line computation? [On-line algorithms would obtain inputs as the time progress. Off-line computation would provide all the possibilities as input at once. Cf. function-to-relation conversion used to fold the output within the input]
24. Would **oracle computing** affect computability? [Oracle computing: A TM with the capability to ask questions to another mechanism]
25. Would **persistent TM** be able to compute more than the standard TM? [Persistent TM: Multiple sessions of TM operation with some memory between them]
26. Would **accelerating** a TM give more power?
27. Would slight **error tolerance** affect any aspect of the Theory of Computation?
28. What would be the ability of a finite automaton with a **queue**?
29. What would be the effect of “**constant**” (as in complexity analysis) in practice?
30. Can any **mathematical function** be represented computationally?
31. What exactly are **power sets** doing to the Theory of Computation?
32. Is what you can do in **logic** the same as what you can do with computation?
33. If you have a research question of **your own**, please consult the instructor first.

// End