

**Philosophy of Computer Science:
An Introductory Course**

William J. Rapaport

**Department of Computer Science and Engineering,
Department of Philosophy, and Center for Cognitive Science
State University of New York at Buffalo, Buffalo, NY 14260-2000**

`rapaport@cse.buffalo.edu`

`http://www.cse.buffalo.edu/~rapaport/`

June 21, 2005

Philosophy of Computer Science:

An Introductory Course

Abstract

There are many branches of philosophy called “the philosophy of X ”, where X = disciplines ranging from history to physics. The philosophy of artificial intelligence has a long history, and there are many courses and texts with that title. Surprisingly, the philosophy of computer science is not nearly as well-developed. This article proposes topics that might constitute the philosophy of computer science and describes a course covering those topics, along with suggested readings and assignments.

1 Introduction

During the Spring 2004 semester, I created and taught a course on the Philosophy of Computer Science. The course was both dual-listed at the upper-level undergraduate and first-year graduate levels and cross-listed in the Department of Computer Science and Engineering (CSE) (where I am an Associate Professor) and the Department of Philosophy (where I have a courtesy appointment as an Adjunct Professor) at State University of New York at Buffalo (“UB”).

The philosophy of computer science is not the philosophy of artificial intelligence (AI); it includes the philosophy of AI, of course, but extends far beyond it in scope. There seem to be less than a handful of such broader courses that have been taught: A Web search turned up some 3 or 4 that were similar to my course in both title and content.¹ There are several more courses with that title, but their content is more accurately described as covering the philosophy of AI. The philosophy of computer science deserves more exposure at the university level. The UB course was popular (with an enrollment of just under 50), and the students found it valuable, not only for its coverage of topics in the philosophy of computer science, but also for the critical-thinking skills they learned (see §4.1). This article presents my ideas on what a course in the philosophy of computer science might look like.

Why teach philosophy of computer science? And why teach it in a computer

science department rather than a philosophy department? As a professor of computer science with a Ph.D. in philosophy (and a previous career as a philosophy professor), I've long been interested in philosophical issues in computer science in general and artificial intelligence in particular. My colleague Stuart C. Shapiro in the UB CSE department had urged me to develop some philosophy courses for our students. Initially, I had resisted this, not being sure that such courses would be acceptable to my department or—more importantly—taken by enough students. Moreover, my colleague Randall R. Dipert in our philosophy department regularly offered an undergraduate course in the philosophy of AI, with which I didn't want to compete.

However, there were many metaphysical, epistemological, and ethical issues that I thought were of interest in the non-AI part of computer science, many of which have only recently begun to be examined in detail by philosophers and philosophically-oriented computer scientists, and many of which shed new light on classical topics in philosophy. This article surveys them and offers some interesting readings that deserve to be more well known. Moreover, a course such as this can serve as an introduction to philosophy for computer science students, an introduction to issues in computer science for philosophy students, a capstone course for senior undergraduate computer science students, or perhaps an overview course for beginning computer-science graduate students.

2 Syllabus

The course syllabus was organized around a set of questions whose various answers we examined during the semester:²

1. **What is philosophy?** In particular, what is “the philosophy of X ” (where X = things like: science, psychology, history, etc.)? [These questions are especially important to discuss in a course primarily aimed at computer science students, who might have misleading ideas of what philosophy is all about—or no idea at all.]

2. **What is computer science?** [Although the “final” answer to this question may simply be the extensional “whatever computer scientists do”, this is a reasonable issue to discuss, even if there is no intensional answer. The following subquestions indicate some of the interesting issues that this main question raises.]
 - (a) What is science? What is engineering?
 - (b) Is computer science a science? Or is it a branch of engineering?
 - (c) If it is a science, what is it a science of?
 - (d) Is it a science of computers (as some authors say)?
 - (e) What, then, is a computer?

- (f) Or is computer science a science of computation (as other authors say)?
- (g) What, then, is computation?
- (h) What is an algorithm? Is an algorithm different from a procedure?
Many authors say that an algorithm is (like) a recipe; is it, or are there important differences?
- (i) What are Church's and Turing's "theses"?
- (j) Some authors claim that there are forms of computation—often lumped together under the rubric "hypercomputation"—that, in some sense, go "beyond" Turing-machine (TM) computation: What is "hypercomputation"?

3. What is a computer program?

- (a) What is the relation of a program to that which it models or simulates?
What is simulation?
- (b) Are programs (scientific) theories?
- (c) What is an implementation?
- (d) What is software? How does it relate to hardware?
- (e) Can (or should) computer programs be copyrighted, or patented?
- (f) Can computer programs be verified?

4. **What is the philosophy of artificial intelligence?**

- (a) What is AI?
- (b) What is the relation of computation to cognition?
- (c) Can computers think?
- (d) What are the Turing Test and the Chinese Room Argument?

5. **What is computer ethics?** [This, like the philosophy of AI, is a vast question, deserving of its own course and having many textbooks devoted solely to it. For my purposes, I decided to focus on questions that don't seem to be the typical ones asked in such a course.]

- (a) Should we trust decisions made by computers?
- (b) Should we build "intelligent" computers?

The remainder of this paper surveys these topics, recommends suggested readings, discusses the sorts of assignments I gave, and presents some student reactions.³

3 Textbooks

Unfortunately, there is no textbook that exactly overlaps the above topics. Three possibilities were offered to the students as recommended texts: Luciano Floridi's *Philosophy and Computing* (1999), Timothy Colburn's *Philosophy and Computer*

Science (2000), and Floridi's *Blackwell Guide to the Philosophy of Computing and Information* (2004). The first two are monographs offering the authors' points of view; there is nothing wrong with this, of course, but I preferred a more neutral approach for the sort of course that I had in mind. Moreover, the topics covered in each of these had a relatively small intersection with my topics. The third book is an anthology, but—again—there was only a small overlap with my topics, and, in any case, I preferred that my students read primary sources rather than overviews.

There are other sources, of course: A special issue of the philosophy journal *The Monist* (Vol. 82, No. 1, 1999) was devoted to the philosophy of computer science. The journal *Minds and Machines: Journal for Artificial Intelligence, Philosophy and Cognitive Science* is almost entirely devoted to philosophy of computer science broadly construed. And about half of the articles in the *Journal of Experimental and Theoretical Artificial Intelligence* are on philosophy of computer science. Finally, an excellent website, "Computational Philosophy", is moderated by John Taylor [<http://www.crumpled.com/cp/>].⁴ In the sections that follow—and more extensively on the course website (see note 3)—I recommend appropriate readings for the topics that we covered.

4 Topics and Readings

4.1 What is philosophy?

A typical advanced philosophy course in a philosophy department normally does not need to address the question of what philosophy is, but I felt that a course whose principal audience was computer-science students needed to. I suspect that many such students feel that philosophy is a “soft” subject where there are no answers, so everyone’s opinion is equally good.⁵ In contrast, I hoped to present to the students a view of philosophy as an analytical and critical discipline that could be of value to them.⁶

I began with a brief history of western philosophy, beginning with Socrates’s and Plato’s view of the philosopher as “gadfly” challenging others’ assumptions. I offered my own definition of philosophy as the search for truth in any field by rational means (which might be limited to deductive logic, or might be extended to include empirical scientific investigation). And we defined the “philosophy of X ” as the study of the fundamental assumptions and main goals of any discipline X .

I briefly covered some of the basic principles of critical thinking and informal argument analysis, including the following notions:

1. “argument” (a set of premises and a conclusion)
2. “premise” (a Boolean proposition used to support a conclusion)

3. “conclusion” (a Boolean proposition that someone tries to convince you of by means of a logical argument)
4. “valid argument” (an argument is valid iff it is impossible for the premises all to be true yet for the conclusion to be false; this semantic notion can also be supplemented with a syntactic one: an argument is (syntactically) valid iff it has the form of any of a given standard set of argument forms that are (semantically) valid, such as Modus Ponens)
5. “factual argument” (this is a non-standard, but useful, notion:⁷ an argument is factual iff all of its premises are true)
6. “sound” (an argument is sound iff it is factual and valid).⁸

I will have more to say about this in §5.2, where I discuss the course assignments, but I should point out that Computing Curricula 2001’s “Social and Professional Issues” knowledge area includes the item “Methods and Tools of Analysis” (SP3), which covers precisely these sorts of argument-analysis techniques [<http://www.computer.org/education/cc2001/final/sp.htm#SP-MethodsAndTools>].

As a reading assignment, I asked the students to read at least one of a variety of brief introductions to philosophy (e.g., Plato’s *Apology*; Colburn 2000, Chs. 3–4; Audi 2001), and I listed Mark B. Woodhouse’s *Preface to Philosophy* (2003) as another recommended textbook for the course.

4.2 What is computer science?

We began the first major section of the course by discussing the reasons one might ask what a discipline is: There are, of course, philosophical—primarily ontological—reasons. But there are also political reasons, especially in the case of a discipline such as computer science, which can be found both in (arts-and-) science faculties as well as in engineering faculties (sometimes in both at the same institution!), or even in its own faculty (either accompanied by other departments in, say, an informatics faculty or else by itself). Then, too, there is the question of the relationship between computer *science* and computer *engineering*.

We surveyed the following answers that have been given to the question “What is computer science?”:

- It is a *science of computers and surrounding phenomena* (such as algorithms, etc.) (Newell et al. 1967).
- It is the *study* (N.B.: not “science”) *of algorithms and surrounding phenomena* (such as the computers they run on, etc.) (Knuth 1974).
- It is the empirical study (“*artificial science*”) of the phenomena surrounding computers (Newell & Simon 1976; cf. Simon 1996).
- It is a *natural science*, not of computers or algorithms, but of *procedures* (Shapiro 2001).

- It is *not* a science, but a branch of *engineering* (Brooks 1996).
- It is the body of knowledge dealing with information-transforming *processes* (Denning 1985).
- It is the study of *information* itself (Hartmanis & Lin 1992).

Note that several of these (especially the first two) might be “extensionally equivalent” but approach the question from very different perspectives: Some emphasize the computer (hardware); others emphasize algorithms, processes, procedures, etc. (software), or even something more abstract (e.g., information). An orthogonal dimension focuses on whether computer science is a science or perhaps something else (a “study”, a “body of knowledge”, an engineering discipline, etc.). And, of course, the name itself varies (computer science, computing science, informatics, etc.), often for political, not philosophical, reasons.⁹

4.2.1 Is “computer science” science or engineering?

The question of whether computer science is really a science or else is really a branch of engineering has been the subject of several essays. It has special relevance at UB ever since our former Department of Computer Science, housed in the Faculty of Natural Sciences and Mathematics, merged with several computer

engineers¹⁰ from our former Department of Electrical and Computer Engineering to form a new Department of Computer Science and Engineering housed in the School of Engineering and Applied Sciences. This is not only confusing to read about, but has given rise to a certain identity crisis for both our students and faculty, and I thought it would provide interesting local color to an investigation of the nature of science and of engineering.

We first turned to the question “What is science?”, discussing both its goals (should it merely *describe* the world—as Ernst Mach thought (cf. Alexander 1967: 118f)—or *explain* it?) as well as the nature of its theories (are they merely instrumentalist, or realist?). We looked at debates over scientific method (is it experimental and cumulative, or does it proceed by paradigm and revolution?) and its branches (is mathematics a science?). The primary readings on science were selections from David Papineau’s “Philosophy of Science” (1996) and chapters from John G. Kemeny’s *A Philosopher Looks at Science* (1959).

We next looked at the history of engineering (Michael Davis’s *Thinking Like an Engineer* (1998), is especially useful), discussing engineering as applied science, as defined in terms of professional education, and as a design activity (Petroski 2003). And we looked at a definition of computer science as a new kind of engineering that studies the theory, design, analysis, and implementation of information-processing algorithms (Loui 1987, 1995).

4.3 What is a computer?—Part I

Insofar as computer science is the science (or study) primarily of computers, the next reasonable question is: What is a computer? This is a large topic, and I divided it into two parts.

The first part was a survey of the history of computers. I presented this in terms of two parallel goals: the goal of building a computing machine, and the goal of providing a foundation for mathematics. As I see it, these were two more-or-less independent goals that converged in the first half of the 20th century. (Whether or not this is a historically accurate way of looking at the matter is itself an interesting question; in any case, it is certainly a convenient way to organize the topic pedagogically.) Our discussion of the first goal involved the contributions of Babbage, Aiken, Atanasoff and Berry, Turing, and Eckert and Mauchly. The contributions of Leibniz, Boole, Frege, Hilbert, Turing, Church, and Gödel made up the overview of the second goal.

The history of computers is a large topic, and we did not spend much time on it. Consequently, the assigned readings were intended only to give the students a flavor of the main events. I prepared a website, “A Very Brief History of Computers”,¹¹ based on the IEEE’s “Timeline of Computing History”¹² and containing links for further information, and I asked the students to read O’Connor & Robertson 1998 (on Babbage), Simon & Newell 1958 (pp. 1–3 are also on

Babbage), and Ensmenger 2004 (on the controversy over who deserved the US patent for the first computer).

4.4 What is an algorithm?—Part I

The other main answer to the question of what computer science studies is: algorithms. So, what is an algorithm? We began our two-part investigation of this by first considering what computation is. One informal, introductory-computer-science-style explanation proceeds as follows: A function f (viewed as a set of ordered pairs, or “inputs” and “outputs”) is *computable* means by definition that there is an “algorithm” that computes f , i.e., there is an algorithm A such that for all input i , $A(i) = f(i)$, and A specifies how f ’s inputs and outputs are related (or how f ’s outputs are produced by its inputs). Then an *algorithm for a problem P* can be characterized as a finite procedure (i.e., a finite set of instructions) for solving P that is:

1. unambiguous for the computer or human who will execute it; i.e., all steps of the procedure must be clear and well-defined for the executor, and
2. effective; i.e., it must eventually halt, and it must output a correct solution to P .¹³

It became an interesting exercise as we went through the semester to compare the different (informal) explications of ‘algorithm’, no two of which seem to be

equivalent. This makes Turing's accomplishment all the more interesting!

With this informal exposition in mind, we then turned to a careful reading of Turing's *magnum opus*, "On Computable Numbers" (1936). There are several versions on the Web, though the most trustworthy is the one reprinted in Davis 1965. When I say "careful reading", I mean it: We spent an entire 80-minute class doing a slow, "active", line-by-line reading of as much of it as we could.¹⁴ I strongly recommend that all computer-science students (as well as computationally-oriented philosophy students, of course) do this at least once in their lives. In addition to being one of the most significant scientific papers of the 20th century, it is also fascinating, well-written, and contains many interesting philosophical insights. The students told me afterwards that this slow reading was one of the highlights of the course.

We also discussed the history of the mathematical investigation of the concept "computable", and discussed the relationship of (1) Turing's thesis that a function is (informally) computable if and only if it is TM-computable to (2) Church's thesis that a function is (informally) computable if and only if it is lambda-definable (which is logically equivalent to being recursive and, of course, to being TM-computable).

Besides Turing 1936, I also asked the students to read Leon Henkin's "Are Logic and Mathematics Identical?" (1962), which has a good discussion of the

history of logic and the foundations of mathematics that led up to Turing's analysis, and Gabor Herman's "Theory of Algorithms" (1983), which discusses the informal notions of "algorithm" and "effective computability" and provides a good background for Turing 1936. I also especially recommend (to instructors, if not to students) Robert I. Soare's "Computability and Recursion" (1996) for the clarity it brings to the history of the competing analyses of 'computable' (e.g., how Turing's Thesis differs from Church's Thesis).

4.5 What is a computer?—Part II

Armed with this background, we turned to philosophical questions surrounding the nature of computers. John Searle's "Is the Brain a Digital Computer?" (1990) argues that *everything* is a digital computer (which seems to trivialize the question), and Patrick Hayes's "What Is a Computer?" (1997) is a symposium that responds to Searle. Hayes's own view is that a computer is a machine that can take, as input, patterns that describe changes to themselves and other patterns, and that causes the described changes to occur. (A related definition—a computer is a device that "change[s] variable assignments"—is offered in Thomason 2003: 328.) It turns out that it is surprisingly difficult to give a precise characterization of what a computer is.

A closely related topic for which a relevant reading did not appear till after

the semester was over is the question of whether the universe itself is a computer (or whether parts of the universe compute; e.g., does the solar system compute Kepler's laws?). On this, see Seth Lloyd & Y. Jack Ng's "Black Hole Computers" (2004). This issue also concerns the nature of simulation (see Rapaport 1998, Perruchet & Vinter 2002 (esp. §1.3.4), and §4.8.2, below).

4.6 What is an algorithm?—Part II

As hard as it is to define 'computer', the notion of "algorithm" is even murkier, despite the accomplishments of Church, Gödel, Kleene, Markov, Turing, Post, et al. Introductions to computer science often liken algorithms to recipes, and, indeed, there are clear similarities. But the differences are even more illuminating, given the informality with which recipes are usually presented. An interesting unpublished paper by Beth Preston (2000) suggests that recipes are more like specifications than they are like algorithms. And Carol Cleland has written a series of papers (1993, 1995, 2001, 2002) that explores the relationships between algorithms, recipes, and procedures, introducing a notion of "mundane" procedures (causal processes, including recipes), which are effective procedures that (she argues) are not TM-computable, since their effectiveness depends on the external world.

4.7 What is hypercomputation?

“Hypercomputation” is a name given by the philosopher Jack Copeland (2002) to the computation of functions that can’t be TM-computed. We briefly investigated Turing’s “oracle” machines, Putnam’s and Gold’s “trial & error” machines (Turing machines where it is the *last* answer that counts, not the first answer), Boolos & Jeffrey’s infinitely-accelerating “Zeus” machines, and Wegner’s “interaction” machines (such as automatic-teller machines or airline-reservation systems) (see Copeland 2002 for citations and other models). We also looked at Kugel’s (2002) thesis that Putnam-Gold machines may be needed for AI to succeed.

4.8 What is a computer program?

We focused on five aspects of this question: the nature of implementation, whether programs are theories, the nature of software (vs. hardware), whether software can or should be copyrighted or patented, and whether programs can be verified. Each is discussed briefly, below, with a digression on course evaluation.

4.8.1 What is implementation?

“Implementation” is a ubiquitous notion in computer science, but one that is rarely defined, and thus crying out for philosophical analysis. We say that programs implement algorithms, yet high-level programs can be implemented in machine

language. We say that particular data structures (e.g., arrays) can implement abstract data types (ADTs) (e.g., stacks), yet some ADTs (e.g., stacks) can be implemented in other ADTs (e.g., linked lists). Is implementation a relation between an abstraction and something “concrete”, or can it (also) be a relation between two abstractions? Is it an isomorphism, or a homomorphism? In rebuttal to Searle’s argument that everything is a computer (see §4.5, above), David Chalmers (1994) develops a notion of implementation as isomorphism. I have urged that implementation is best viewed as the semantic interpretation of an abstract formal system (Rapaport 1999 and forthcoming-b). These issues were all touched upon, and I also used this opportunity to carefully develop the notions of syntax, semantics, and formal systems.

4.8.2 Are programs scientific theories?

Some computational cognitive scientists (e.g., Pylyshyn 1984: 76, Johnson-Laird 1988: 52) have claimed that cognitive theories are best expressed, not in the languages of statistics or mathematics, or even in natural language, but in computer programs. These programs, being simultaneously theories and models (or implementations of the theories), can then be executed, in order to test whether the theory is a good model of cognition. It has also been argued, of course, that such a program is more than merely a model or simulation of the cognitive phenomenon

under investigation; some have argued that it actually exhibits the cognitive ability. As background, we also discussed the relationships between theories and models, simulations and “the real thing”, and simulations and emulations; philosophical theories of scientific explanation; and philosophical theories of scientific models. Relevant readings here also include Joseph Weizenbaum’s *Computer Power and Human Reason* (1976; Chs. 5 and 6 are on models and theories) and Herbert Simon’s *Sciences of the Artificial* (1996; Ch. 1, which discusses scientific theories, is also good reading for the question of whether computer science is a science).

4.8.3 What is software?

Introductory computer science courses often assume that the distinction between software and hardware is clear. Computer scientists and philosophers know otherwise. James Moor’s “Three Myths of Computer Science” (1978) points out the inadequacies of the usual “abstract” software vs. “concrete” hardware distinction, arguing that software is a computer program that is changeable by a person. This allows for the “software” to be “hard”wired, as long as it can be changed. The “software is abstract” point of view is well argued by Peter Suber (1988), who considers it to be “syntactic form” (and this ties in nicely with the discussion of syntax vs. semantics in the section on implementation). Finally, Colburn (1999) views software as a “concrete abstraction”: It has a “medium of

description” insofar as it is a text in a formal language (which is an abstraction), and it has a “medium of execution” insofar as it is implemented in circuits and semiconductors (which are concrete).

4.8.4 Interlude: Midsemester Course Evaluation and Course Correction

The previous topic brought us more or less to the midsemester point in the course. Borrowing an idea from my colleague Stuart C. Shapiro, I traditionally give a midsemester course evaluation. I strongly recommend this for any course: It is far more useful than an end-of-course evaluation that is not seen until the course is over and hence is of no use in improving the course that just ended. For this course, I asked two simple, open-ended questions: What aspects of the course would you like to see changed? and What aspects of the course do you especially like? The answers let me know what needed to be fixed and what was going well. I summarized the answers and posted a response to the course newsgroup.

For this course, the major complaint was the amount of reading. I told the students that I would try to comply with their request for less reading, but that there were just so many exciting things that I wanted them to read that I would compromise: From then on, I only assigned one (sometimes two) required readings for each of the remaining topics, per class session, but I recommended (sometimes strongly) other things to look at—if not now, then at their leisure after the semester

was over. Thus, for example, instead of requiring the students to read Moor 1978 *and* Suber 1988 (which is a very long paper) *and* Colburn 1999 (which is philosophically challenging), I only *required* them to read Moor 1978 (which is well-written and also discusses other important topics), *strongly recommended* Suber 1988 (which is wide-ranging and has lots of things to think about), and recommended Colburn 1999. In lecture, however, I discussed all three.

I hasten to add that there were many compliments, too! Students were pleased with the topics and organization, and especially liked the writing assignments, which I discuss further in §5.2.

4.8.5 Can software be patented? Or should it be copyrighted?

The topic of whether computer programs are copyrightable entities or patentable entities¹⁵ is a fascinating one, because it combines legal, social, and metaphysical issues. We concentrated on the last of these, since it flows nicely from the previous topic of what software is.

Here is the fundamental paradox: If a computer program is viewed as a written text, then it is, by definition, copyrightable. But the very “same” program, engraved on a CD-ROM and, hence, executable on a computer, can be viewed as a machine that is, by definition, patentable (as well as subject to legal limitations on exportation to foreign countries; see Colburn 1999). Yet, also by definition, nothing

is both copyrightable and patentable. (Whether one *should* copyright or patent a program vs. whether programs should be “open source” is one of the interesting social issues that we did not have time to consider.)

We looked at the legal definitions of copyright and patent (available from various US government websites)¹⁶ and read a fascinating—and little known—essay by computer scientist Allen Newell (“The Models are Broken, the Models are Broken”) that appeared as part of a symposium on this topic in the *University of Pittsburgh Law Review* (1985-1986). Newell argues that we computer scientists need to devise better models—i.e., better ontological theories—of such computer-science entities as algorithms, programs, etc. In contrast, some legal scholars (e.g., Koepsell 2000) have argued that lawyers need to devise better methods of legal protection that better match the unique natures of computer software and hardware. The point in both cases is that there is a mismatch between computer-science entities, on the one hand, and legal forms of protection, on the other (or between computational ontology and legal ontology); something’s got to give.

4.8.6 Can programs be verified?

We ended our investigations into the nature of computer programs with an inquiry into whether they can be formally verified. There is a subfield of computer science and software engineering that looks into formal methods for proving program

correctness (see, e.g., Gries 1981 for a classic treatment). Two philosophers have written essays that critique this approach. I am a firm believer in the value of such formal proofs (despite some very real limitations), and I have several times taught our department's course on program verification. Consequently, I spent some time introducing some aspects of formal verification before turning to the criticisms.

Brian Cantwell Smith's (1985) "Limits of Correctness in Computers" is, in my opinion, one of the most significant papers on all aspects—moral, legal, semantic, ontological, etc.—of the philosophy of computer science, and should be required reading for all computer science majors. Among other things, he argues that there is a gap between the world and our models of it and that computers are doubly removed, relying on models of the models, yet must act in the real world.

The other critique is James Fetzer's explosive essay, "Program Verification: The Very Idea", that appeared in the *Communications of the ACM* in 1988 and that launched a vicious public debate on the pros and cons of verification. Briefly, Fetzer argues that *programs* can't be verified because you can't logically prove that causal systems won't fail; at best, you can verify an *algorithm*. Note that, in order to properly evaluate Fetzer's argument, you must have a firm grasp of the relationship of algorithm to program, which, by this time, my students were well-prepared for.

4.9 Philosophy of AI: *Could* we build artificial intelligences?

As I indicated above, the philosophy of AI deserves a full course to itself (see, e.g., Moulton & Voytek 1979, Rapaport 1986), and one of my motivations for creating a course in the philosophy of computer science (and not merely the philosophy of AI) was that there were many non-AI philosophical issues of interest. Nevertheless, the philosophy of AI is a proper part of the philosophy of computer science, it is my own area of expertise, and the students intensely wanted to discuss it.

I limited myself to two topics: the Turing Test and the Chinese-Room Argument. A case can be made that an excellent course on the philosophy of computer science could consist solely of close readings of Turing's two major essays: his 1936 paper on computability and his 1950 paper on whether computers can think. So, for this topic, we read Turing's "Computing Machinery and Intelligence" (1950) as well as the current (and probably perennially most popular) reply: John Searle's Chinese-Room Argument ("Minds, Brains, and Programs", 1980).

Turing 1950, as is well known, argued that a computer will be said to be able to think if we cannot distinguish its linguistic (hence cognitive) behavior from a human's. Searle 1980 proposed a now-classic counterexample that alleges that a computer could pass a Turing Test without really being able to think.¹⁷ (A good source for both of these, and related, papers is Shieber 2004; cf. Rapaport,

forthcoming-a.) We closed this topic with my own attempt at a rebuttal of Searle (Rapaport 2000), arguing that syntactic symbol manipulation of the sort that computers do can suffice for semantic interpretation of the kind needed for computational cognition.

4.10 Computer ethics

Our final topic was computer ethics. As noted above, and as with philosophy of AI, this is often the topic of full courses by itself and is the subject of numerous texts and anthologies. I gave a brief overview of (computer) ethics, based on Moor's "What Is Computer Ethics?" (1985). We focused on his claim that we need to have metaphysical and ontological theories of computers (in particular, their "logical malleability") and related phenomena in order to answer ethical and social questions about their nature and use.

I chose to concentrate on two issues that are not often covered in such courses or books: Are there decisions that computers should never make? and Should we build artificial intelligences?

We turned to Moor's "Are There Decisions Computers Should Never Make?" (1979). One of his main points is that there are no decisions computers shouldn't make, at least as long as their track record is better than that of humans, but it's up to us to accept or reject their decisions. An interesting contrasting opinion is

that of Friedman & Kahn’s “People Are Responsible, Computers Are Not” (1992), which argues that there *are* decisions that computers should not make, because only humans are capable of being moral agents. But “to err is human”, and we looked at a recent case of an airline crash caused by following a human’s decision instead of a computer’s (as reported in George Johnson’s “To Err Is Human”, 2002).

On ethical issues in AI, we read Michael R. LaChat’s “Artificial Intelligence and Ethics: An Exercise in the Moral Imagination” (1986). First, I outlined the plot of Stanislaw Lem’s “Non Serviam” (1971)—which should be required reading for all researchers in artificial life!—in which what we would today call an Artificial Life researcher is forced to pull the plug on his creations when his research grant ends. LaChat considers whether such research shouldn’t even *begin* but that, nevertheless, considering the possibilities enables us to deal with important issues such as: What is a person? Would an AI with personhood have rights? Could it be moral?

4.11 Philosophy of computer science: A summary and a unifying theme

In closing the semester, I asked the students to read two recent overviews of issues in the philosophy of computer science, as a way to gauge what they had learned: Matthias Scheutz’s “Philosophical Issues about Computation” (2002) and Smith’s

“The Foundations of Computing” (2002), and we reviewed the semester’s readings and discussion, with an eye towards themes that connected the several topics.

One such theme that the students and I became aware of as the semester progressed is the relation of an abstract computation to the real world. This theme is addressed explicitly in some of the papers we read, and is implicit in many others. It emerges in Cleland’s discussion of the causal nature of “mundane” procedures, which produce some actual product or physically affect the real world in some way. This is also one of Smith’s themes in his “Limits of Computation” essay, as well as an underlying reason of Fetzer’s arguments against program verification. It is, of course, the subject matter of implementation, and underlies the paradoxical nature of software vs. hardware, and hence the issue of whether software is copyrightable or patentable. I recommend an exploration of this theme as a unifying idea for a future course in philosophy of computer science.

5 Assignments

5.1 A difficulty

I wanted the students to do a lot of reading and thinking. Thinking is best done by active reading (Rapaport 2005a), discussion, and writing—lots of writing. There is a well-known drawback to assigning a lot of writing to students: The

instructor has to read it all and, ideally, comment on it. When this course was first advertised, I expected about 10–15 students, in a small seminar-like setting. The first preliminary enrollment report said that 30 students had signed up. Thinking that they thought that this might be a “gut” course (“A philosophy course in a computer science department? Oh, this’ll be easy to ace!”), I posted a note to the undergraduate newsgroup spelling out the large quantities of writing that I would expect. Enrollment doubled to 60! It finally settled down at just under 50 students.¹⁸ Still, 50 ten-page term papers plus frequent short writing assignments during the semester was not a prospect that I looked forward to.

Nor could I rely on help from graduate teaching assistants or recitation sections (a problem I was familiar with from my days teaching at a primarily undergraduate institution). No recitation sections had been assigned to the course, since I had not expected such a large enrollment. They would have been useful for discussion purposes, but that was not to be. I did have an excellent graduate teaching assistant, but he was a computer-science grad student, not a philosophy grad student (although he did have some undergraduate philosophy experience and was philosophically sophisticated), and, in any case, he had no recitation sections to lead. Consequently, he was of most use to me in keeping records, though he did hold office hours and students felt comfortable going to him for advice on writing.

But how to have students write a lot without having to read it all? And how

to have discussions without special time allotted for them? Of course, faculty at undergraduate institutions face this problem all the time, unlike we faculty at research universities. And so I drew upon my experiences as a philosophy professor at an undergraduate institution with no TAs and no recitation sections.

5.2 A solution: Required, short position papers . . .

I assigned the students five 1-page position papers throughout the semester, roughly one every 2 or 3 weeks. A first draft of each assignment was due 1 week after it was announced. The day it was due we set aside for “peer editing” (adapted from techniques used in English composition classes; cf. Cho & Schunn 2004): Each student was asked to bring 5 copies of their position paper, one for me, one for themselves, and one each for 3 other students. I put the students into small groups of three or four “peers”, each of whom had written a response to the same assignment. I asked them to spend about 10–15 minutes on each paper, reading it, critiquing it, and making suggestions for improvement. The students were then given another week to revise their papers to be handed in for credit.¹⁹ To ease my burden of grading, I read and put copious comments on only about 40% of the papers for each of the 5 assignments; each student received at least 2 papers fully critiqued by me (the other 3 papers were recorded as being handed in).

Peer editing accomplished several goals simultaneously: The students had

plenty of opportunities to discuss the material with each other. In fact, probably more students participated in these small groups than would have ordinarily spoken out in a large classroom setting (though such full-class discussions were encouraged, as well). Moreover, all students got multiple feedback on each paper, in addition to my feedback on a subset of their papers. Another advantage of peer editing in class is that I had the freedom (and responsibility) to walk around the room, listening to the student discussions and occasionally facilitating one or answering a question on a one-on-one basis.

The position papers were designed to supplement the lectures and readings, as well as to foster the students' critical-thinking skills. In particular, the topics always involved an argument that the students were asked to evaluate in terms of factuality (i.e., truth value of the premises) and validity (i.e., reasoning). Students were encouraged to present their opinions and to support them with reasons. As one example, Position Paper 1, on "What is computer science?", asked the students to evaluate the following argument (embedded in the context of a story about a dean moving a computer science department from a school of science to a school of engineering):

1. Science is the systematic observation, description, experimental investigation, and theoretical explanation of natural phenomena.
2. Computer science is the study of computers and related phenomena.

3. Therefore, computer science is not a science.

(All assignments and peer-editing guidelines are on the Web at [<http://www.cse.buffalo.edu/~rapaport/510/pospapers.html>].)

As one student observed later, the argument-analysis format of the position papers made them somewhat easier to grade than an ordinary essay would have been. Since the students were required to examine a rigid structure of an argument, they had fewer “degrees of freedom” in writing their responses. Thus, grading such papers can be closer to grading a mathematical problem set than a typical essay. It also made grading somewhat more impartial and somewhat less controversial.²⁰

5.3 ... And two optional assignments

In addition to the required position papers, there was an optional term paper, whose topic had to be approved by me in advance. I supplied a list of some possible topics, but I encouraged the students to explore areas of interest to them. As a default topic, a student could write a summary of the philosophy of computer science in the style of an encyclopedia article or else present his or her own answers to the syllabus questions (see §2).

An exclusive-alternative option was a final exam (students could do the exam or the term paper, but not both). This was a take-home, short-answer, essay-style exam, asking for analytic and evaluative summaries of the possible answers to the

topic-questions.

5.4 A required reading journal

In addition, in order to provide evidence that the students were really reading the material, as well as to encourage them to read slowly and actively, I required them to keep a “reading journal”. For each essay they read, they were to copy interesting passages (or at least full references to them) and—most importantly—to provide their own comments on them and on the issues raised in each item read. (Suggestions on how to do this can be found in Rapaport 2005a.) I collected these Journals at the end of the semester, and included them in the grade calculation.

Students who attended almost all classes and turned in a Reading Journal could get a C; students who did that plus all five position papers could get a B; and students who did all of that plus either the term paper or final exam could get an A. All but one student wrote the position papers. Over 80% of the students chose the exam/paper option, with about 70% choosing the exam option.

6 What the students did and didn’t like

The students’ favorite part of the course was the writing, peer-editing, and revising of the 1-page position papers: They enjoyed the discussions, the ability to revise (including an option to re-revise for a higher grade), and—most importantly—the

skills they learned, and the practice they got, in critically analyzing and evaluating informal arguments. In addition, well after the course ended, some students told me that they have continued keeping reading journals in their other courses and research.

They also appreciated the course website, which has links to the syllabus and a directory of documents that, in turn, has a large bibliography, links to other relevant websites, and links to the assignments, position papers, term-paper topics, and final exam. I began each new section of the course by putting up a webpage containing recommended readings for that topic. I then gave a quick overview in lecture about each of the readings. Students informed me that this was very useful because it provided a summary of what was to come, including the different positions that have been taken on each issue.

Here is what one student said, in an unsolicited email message I received after the course was over:

I'd like to thank you for putting together such a great course this semester. I'll admit, knowing very little about it, I never had much respect for philosophy in the past—but this course has provided me with an entirely new perspective. In fact, I'd say that I learned as much in your course as any other I've taken in my graduate career at UB (not to mention the fact that the skills I learned in [it] are far more

transferable than the skills of the more esoteric CS courses). . . . I urge [you] to offer this course again in the future. It offers exactly the kind of breadth of education that the department needs to stress, and with its CS flavor, it can tap the interest of students who would otherwise blow it off. Thanks again for a great semester, and please consider making Philosophy of CS a regular offering :)

Another student observed that “I don’t think there was a single student in the class whose critical thinking/writing/reading skills didn’t improve as a result of taking this course.”

As noted above, the students’ least favorite part of the course was the amount of reading. Of course, this is something that students almost always complain about, but, in this case, the complaint really was about the quantity, not the quality: By and large, they found all of the readings to be interesting and useful; their complaint was that they didn’t have enough time to read them all as carefully as they (and I) would have liked. Fortunately, on the basis of the midsemester course evaluation, I found this out early enough to be able to do something about it. As discussed above, subsequent reading assignments were limited to at most two required readings, with suggestions for recommended (but optional) follow-up readings.

7 Conclusions

I believe this to have been a worthwhile course, both for me and—more importantly—for the students. It gave many of the computer science majors an option to think about many issues that they either hadn't thought of before or had thought about but had no venue for discussing. It also gave them an opportunity to (learn how to) think critically, and to find out what philosophy could be like. The philosophy majors, in addition, had the opportunity to learn some things about computers, computing, and computer science that they probably would not have come across in more traditional philosophy courses, as well as the opportunity to apply some of their philosophical skills and knowledge to a different domain.²¹

NOTES

¹In particular, CD5650, Swedish National Course on Philosophy of Computer Science, at Mälardalen University (Sweden), coordinated by Gordana Dodig-Crnkovic [<http://www.idt.mdh.se/~gdc/PI-network-course.htm>]; Selected Topics in the Philosophy of Computer Science, at Tel Aviv University (Israel), taught by Eli Dresner [<http://www.tau.ac.il/humanities/digicult/english.htm>]; and PHI 319, Philosophy of Computing, at Arizona State University, taught by Bernard W. Kobes [http://www.asu.edu/clas/philosophy/course_descripts.htm].

²I am grateful to Timothy Colburn, Randall R. Dipert, Eli Dresner, James H. Fetzer, Luciano Floridi, Bipin Indurkha, James Moor, Robert Stainton, and Chris Viger for (email) discussions on the questions that such a course might focus on.

³The homepage for the course, with links to the complete syllabus, assignments, and other course webpages, is at [<http://www.cse.buffalo.edu/~rapaport/philcs.html>] and archived as Rapaport 2005b.

⁴Pointers to these and other sources are at my course webpage “What is Philosophy of Computer Science?” [<http://www.cse.buffalo.edu/~rapaport/510/whatisphilcs.html>].

⁵This claim is based on a not unreasonable assumption that computer-science students tend to be “Dualists” who see (and fear?) philosophy as being a “Multiplistic” discipline. These are terms from William Perry’s (1970, 1981) “scheme” of intellectual and ethical development. For a quick online glimpse of Perry’s views, see my website, “William Perry’s Scheme of Intellectual and Ethical Development” [<http://www.cse.buffalo.edu/~rapaport/perry.positions.html>]. Roughly, “Dualists” believe that all questions have correct answers and that the student’s job is to learn these answers, whereas “Multiplists” believe that most questions have no known answers and, consequently, that everyone’s opinion is equally good. However, those are vast oversimplifications, and the interested reader is urged to consult Perry’s writings, or any of the other sources listed on my website. On whether there can be answers to philosophical questions and, thus, real progress in philosophy, see Rapaport 1982.

⁶In Perry’s terminology, philosophy is a “Contextually Relativistic” discipline, i.e., one that critically evaluates claims on the basis of evidence (the truth-value of a claim is “relative” to its evidential “context”).

⁷Learned from my former philosophy colleague, Kenneth G. Lucey.

⁸There are many excellent textbooks on critical thinking and informal logic, and, of course, it is the subject of many full courses on its

own. A useful short introduction for a course such as this is Longview Community College's website "Critical Thinking Across the Curriculum Project" [<http://www.kcmetro.cc.mo.us/longview/ctac/toc.htm>].

⁹Another (more recent) view is that computer science is the study of *virtual* phenomena (Crowcroft 2005).

¹⁰Some of whom had their doctorates from departments of computer *science*!

¹¹[<http://www.cse.buffalo.edu/~rapaport/510/history.html>]

¹²[<http://www.computer.org/computer/timeline/>]

¹³This is an adaptation of Stuart C. Shapiro's informal characterization; personal communication.

¹⁴On this sort of Talmudic, slow-but-active reading style, see Rapaport 2005a, §5.

¹⁵There is a third possibility: that they are trademarkable entities; we did not consider this option.

¹⁶For 'copyright', see the US Copyright Office Circular 1 at [<http://www.copyright.gov/circs/circ1.html#wci>];

for 'patent', see the US Patent and Trademark Office Glossary at [<http://www.uspto.gov/main/glossary/index.html#p>].

¹⁷Mention should be made that a very early version of Searle's thought experiment appears as a way of explicating Turing machines in Rogers 1959 (Part

I, reprinted as Rogers 1969: 131, 133; based on a 1957 lecture).

¹⁸The breakdown of student majors was as follows:

	CSE	PHI	Other
undergrads	72%	10%	12%
grads	75%	15%	10%
total	73%	12%	15%

CSE = Computer Science & Engineering majors; PHI = Philosophy majors; Other = students majoring in Biology, Economics, Electrical Engineering, Management, Management & Information Science, and Mathematics.

¹⁹My writing guidelines and a brief guide to grammar and punctuation are on the Web at [<http://www.cse.buffalo.edu/~rapaport/howtowrite.html>].

²⁰For more thoughts on grading, and a “triage” theory of grading, see my website, “How I Grade” [<http://www.cse.buffalo.edu/~rapaport/howiggrade.html>].

²¹I am grateful to my students Dima Dligach and Albert Goldfain, to my colleagues Peter D. Scott and Stuart C. Shapiro, and to an anonymous reviewer for comments on earlier drafts.

References

- Alexander, Peter (1967), "Mach, Ernst", in Paul Edwards (ed.), *Encyclopedia of Philosophy* 5: 115–119.
- Audi, Robert (2001), "Philosophy: A Brief Guide for Undergraduates"
[<http://www.apa.udel.edu/apa/publications/texts/briefgd.html>].
- Brooks, Frederick P., Jr. (1996), "The Computer Scientist as Toolsmith II",
Communications of the ACM 39(3) (March): 61–68.
- Chalmers, David J. (1994), "On Implementing a Computation", *Minds and Machines* 4
(1994): 391-402.
- Cho, Kwangsu; & Schunn, Christian D. (2004), "You Write Better When You Get
Feedback from Multiple Peers than an Expert", *Proceedings of the 20th Annual
Conference of the Cognitive Science Society* (Mahwah, NJ: Lawrence Erlbaum
Associates, 2005).
- Cleland, Carol E. (1993), "Is the Church-Turing Thesis True?", *Minds and Machines* 3(3)
(August): 283–312.
- Cleland, Carol E. (1995), "Effective Procedures and Computable Functions", *Minds and
Machines* 5(1): 9–23.
- Cleland, Carol E. (2001), "Recipes, Algorithms, and Programs", *Minds and Machines*
11(2) (May): 219–237.
- Cleland, Carol E. (2002), "On Effective Procedures", *Minds and Machines* 12(2) (May):
159–179.
- Colburn, Timothy R. (1999), "Software, Abstraction, and Ontology", *The Monist* 82(1):

3–19; reprinted (in slightly different form) in Colburn 2000, Ch. 12.

Colburn, Timothy R. (2000), *Philosophy and Computer Science* (Armonk, NY: M.E. Sharpe).

Copeland, B. Jack (2002), “Hypercomputation”, *Minds and Machines* 12(4): 461–502.

Davis, Martin (ed.) (1965), *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions* (New York: Raven Press).

Davis, Michael (1998), *Thinking Like an Engineer: Studies in the Ethics of a Profession* (New York: Oxford University Press).

Denning, Peter J. (1985), “What Is Computer Science?”, *American Scientist* 73 (January–February): 16–19.

Ensmenger, Nathan (2004), “Bits of History: Review of A.R. Burks’s *Who Invented the Computer? The Legal Battle that Changed Computing History*”, in *American Scientist* 91 (September–October): 467–468.

Fetzer, James H. (1988), “Program Verification: The Very Idea”, *Communications of the ACM* 31(9) (September): 1048–1063; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 321–358.

Floridi, Luciano (1999), *Philosophy and Computing: An Introduction* (London: Routledge).

Floridi, Luciano (2004), *The Blackwell Guide to the Philosophy of Computing and Information* (Malden, MA: Blackwell).

Friedman, Batya; & Kahn, Peter H., Jr. (1992), “People Are Responsible, Computers Are

Not”, excerpt from their “Human Agency and Responsible Computing: Implications for Computer System Design”, *Journal of Systems Software* (1992): 7–14; excerpt reprinted in M. David Ermann, Mary B. Williams, & Michele S. Shauf (eds.) *Computers, Ethics, and Society, Second Edition* (New York: Oxford University Press, 1997): 303–314.

Gries, David (1981), *The Science of Programming* (New York: Springer-Verlag).

Hartmanis, Juris, & Lin, Herbert (1992), “What Is Computer Science and Engineering?”, in Juris Hartmanis & Herbert Lin (eds.), *Computing the Future: A Broader Agenda for Computer Science and Engineering* (Washington, DC: National Academy Press), Ch. 6, pp. 163–216.

Hayes, Patrick J. (1997), “What Is a Computer? An Electronic Discussion”, *The Monist* 80(3).

Henkin, Leon (1962), “Are Logic and Mathematics Identical?”, *Science* 138(3542) (November 16): 788–794.

Herman, Gabor T. (1983), “Algorithms, Theory of”, in Anthony S. Ralston (ed.), *Encyclopedia of Computer Science and Engineering, 2nd edition* (New York: Van Nostrand Reinhold): 57–59.

Johnson, George (2002), “To Err Is Human”, *New York Times* (14 July).

Johnson-Laird, Philip N. (1988), *The Computer and the Mind: An Introduction to Cognitive Science* (Cambridge, MA: Harvard University Press).

Kemeny, John G. (1959), *A Philosopher Looks at Science* (Princeton: D. van Nostrand).

Koepsell, David R. (2000), *The Ontology of Cyberspace: Philosophy, Law, and the Future of Intellectual Property* (Chicago: Open Court).

- Knuth, Donald (1974), "Computer Science and Its Relation to Mathematics", *American Mathematical Monthly* 81(4) (April): 323–343.
- Kugel, Peter (2002), "Computing Machines Can't Be Intelligent (. . . and Turing Said So)", *Minds and Machines* 12(4): 563–579.
- LaChat, Michael R. (1986), "Artificial Intelligence and Ethics: An Exercise in the Moral Imagination", *AI Magazine* 7(2): 70–79.
- Lem, Stanislaw (1971), "Non Serviam", in S. Lem, *A Perfect Vacuum*, trans. by Michael Kandel (New York: Harcourt Brace Jovanovich, 1979).
- Lloyd, Seth, & Ng, Y. Jack (2004), "Black Hole Computers", *Scientific American* 291(5) (November): 52–61.
- Loui, Michael C. (1987), "Computer Science Is an Engineering Discipline", *Engineering Education* 78(3): 175–178.
- Loui, Michael C. (1995), "Computer Science Is a New Engineering Discipline", *ACM Computing Surveys* 27(1) (March): 31–32.
- Moor, James H. (1978), "Three Myths of Computer Science", *British Journal for the Philosophy of Science* 29(3) (September): 213–222.
- Moor, James H. (1979), "Are There Decisions Computers Should Never Make?", *Nature and System* 1: 217–229.
- Moor, James H. (1985), "What Is Computer Ethics?", *Metaphilosophy* 16(4) (October): 266–275.
- Moulton, Janice, & Voytek, Jane (1979), "Can Humans Think Machines Think?", *Teaching Philosophy* 3(2): 153–167.
- Newell, Allen (1985-1986), "Response: The Models Are Broken, the Models Are Broken",

University of Pittsburgh Law Review 47: 1023–1031.

Newell, Allen; Perlis, Alan J.; & Simon, Herbert A. (1967), “Computer Science”, *Science* 157(3795) (22 September): 1373–1374.

Newell, Allen, & Simon, Herbert A. (1976), “Computer Science as Empirical Inquiry: Symbols and Search”, *Communications of the ACM* 19(3) (March): 113–126.

O’Connor, J.J., & Robertson, E.F. (1998), “Charles Babbage”

[<http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Babbage.html>].

Papineau, David (1996), “Philosophy of Science”, in Nicholas Bunnin & E.P. Tsui-James (eds.), *The Blackwell Companion to Philosophy* (Oxford: Blackwell): 290–324.

Perry, William G., Jr. (1970), *Forms of Intellectual and Ethical Development in the College Years: A Scheme* (New York: Holt, Rinehart, and Winston).

Perry, William G., Jr. (1981), “Cognitive and Ethical Growth: The Making of Meaning”, in Arthur W. Chickering and Associates, *The Modern American College* (San Francisco: Jossey-Bass): 76–116.

Perruchet, Pierre, & Vinter, Annie (2002), “The Self-Organizing Consciousness”, *Behavioral and Brain Sciences* 25(3) (June): 297–388.

Petroski, Henry (2003), “Early Education”, *American Scientist* 91 (May-June): 206–209.

Preston, Beth (2000), “Recipes and Songs: Towards a Theory of Production” (unpublished ms.)

Pylyshyn, Zenon W. (1984), *Computation and Cognition: Toward a Foundation for Cognitive Science* (Cambridge, MA: MIT Press).

Rapaport, William J. (1982), “Unsolvable Problems and Philosophical Progress”, *American Philosophical Quarterly* 19: 289–298.

- Rapaport, William J. (1986), "Philosophy of Artificial Intelligence: A Course Outline", *Teaching Philosophy* 9: 103–120.
- Rapaport, William J. (1998), "How Minds Can Be Computational Systems", *Journal of Experimental and Theoretical Artificial Intelligence* 10: 403–419.
- Rapaport, William J. (1999), "Implementation Is Semantic Interpretation", *The Monist* 82(1): 109-130.
- Rapaport, William J. (2000), "How to Pass a Turing Test: Syntactic Semantics, Natural-Language Understanding, and First-Person Cognition", *Journal of Logic, Language, and Information*, 9(4): 467–490; reprinted in James H. Moor (ed.), *The Turing Test: The Elusive Standard of Artificial Intelligence* (Dordrecht: Kluwer, 2003): 161–184.
- Rapaport, William J. (2005a), "How to Study"
[<http://www.cse.buffalo.edu/~rapaport/howtostudy.html>].
- Rapaport, William J. (2005b), "Philosophy of Computer Science: An Introductory Course", *Technical Report 2005-16* (Buffalo: SUNY Buffalo Department of Computer Science & Engineering); pre-print version of this article, containing archived webpages; available at
[<http://www.cse.buffalo.edu/tech-reports/2005-16.pdf>].
- Rapaport, William J. (forthcoming-a), Review of Shieber 2004, *Computational Linguistics*.
- Rapaport, William J. (forthcoming-b), "Implementation Is Semantic Interpretation: Further Thoughts", *Journal of Theoretical and Experimental Artificial Intelligence*.
- Rogers, Hartley, Jr. (1959), "The Present Theory of Turing Machine Computability", *Journal of the Society for Industrial and Applied Mathematics* 7: 114–130; reprinted in Jaakko Hintikka (ed.), *The Philosophy of Mathematics* (London: Oxford University

Press, 1969): 130–146.

Scheutz, Matthias (2002), “Philosophical Issues about Computation”, *Encyclopedia of Cognitive Science* (London: Macmillan).

Searle, John R. (1980), “Minds, Brains, and Programs”, *Behavioral and Brain Sciences* 3: 417–457.

Searle, John R. (1990), “Is the Brain a Digital Computer?”, *Proceedings and Addresses of the American Philosophical Association* 64: 21–37.

Shapiro, Stuart C. (2001), “Computer Science: The Study of Procedures”

[<http://www.cse.buffalo.edu/~shapiro/Papers/whatiscs.pdf>].

Shieber, Stuart M. (2004), *The Turing Test: Verbal Behavior as the Hallmark of Intelligence* (Cambridge, MA: MIT Press).

Simon, Herbert A. (1996), *The Sciences of the Artificial, Third Edition* (Cambridge, MA: MIT Press).

Simon, Herbert A., & Newell, Allen (1958), “Heuristic Problem Solving: The Next Advance in Operations Research”, *Operations Research* 6(1) (January–February): 1–10.

Smith, Brian Cantwell (1985), “Limits of Correctness in Computers”, *Technical Report CSLI-85-36* (Stanford, CA: Center for the Study of Language and Information); first published in Charles Dunlop & Rob Kling (eds.), *Computerization and Controversy* (San Diego: Academic Press, 1991): 632–646; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 275–293.

- Smith, Brian Cantwell (2002), "The Foundations of Computing", in Scheutz, Matthias (ed.), *Computationalism: New Directions* (Cambridge, MA: MIT Press): 23–58.
- Soare, Robert I. (1996), "Computability and Recursion", *Bulletin of Symbolic Logic* 2(3) (September): 284–321.
- Suber, Peter (1988), "What Is Software?", *Journal of Speculative Philosophy* 2(2): 89–119.
- Thomason, Richmond H. (2003), "Dynamic Contextual Intensional Logic: Logical Foundations and an Application", in P. Blackburn et al. (eds.), *CONTEXT 2003*, Lecture Notes in Artificial Intelligence 2680 (Berlin: Springer-Verlag): 328–341.
- Turing, Alan M. (1936), "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, Ser. 2, Vol. 42: 230–265.
- Turing, Alan M. (1950), "Computing Machinery and Intelligence", *Mind* 59: 433–460; reprinted in Shieber 2004.
- Weizenbaum, Joseph (1976), *Computer Power and Human Reason: From Judgment to Calculation* (New York: W.H. Freeman).
- Woodhouse, Mark B. (2003), *A Preface to Philosophy, 7th edition* (Belmont, CA: Wadsworth/Thomson Learning).

UNIVERSITY AT BUFFALO - STATE UNIVERSITY OF NEW YORK
The Department of Computer Science & Engineering

cse@buffalo

CSE 410/510 & PHI 498: PHILOSOPHY OF COMPUTER SCIENCE Spring 2004



SYLLABUS

(This is a living document; the latest version will always be available on the Web at:

<http://www.cse.buffalo.edu/~rapaport/510/syl.html>)

Last Update: 20 April 2004

Note: **NEW** or **UPDATED** material is highlighted



Index:

- [Course Description](#)
- [Prerequisites](#)
- [Staff](#)
- [Class Meetings](#)
- [Texts](#)
- [Important Dates & Tentative Schedule](#)
- [Reading](#)
- [Attendance, Homeworks, Projects, Exams, Newsgroup](#)
- [Homeworks](#)

Other Relevant Links:

- [Homepage](#)
- [Directory of Documents](#)
- [Newsgroup Archive](#)

- [Projects](#)
 - [How to Study](#)
 - [Grading](#)
 - [Incompletes](#)
 - [Academic Integrity](#)
 - [Classroom Disruptions](#)
-

COURSE DESCRIPTION:

This is a new course at UB and a relatively new course anywhere. (I have found only one or two courses anywhere that cover similar material; most courses named "Philosophy of Computer Science" are really about whether computers can think. We'll cover that, but only as one among many other topics.)

I hope to investigate proposed answers to the following questions:

1. What is philosophy?

And, in particular, what is "the philosophy of X" (where X = things like: science, psychology, history, etc.)?

2. What is computer science?

To answer this, we'll need to consider questions such as: What is science? Is computer science a science? If so, what is it a science of? Is it a science of *computers*? What *is* a computer? Is it a science of *computation*? What *is* computation? Computations are said to be algorithms, so what is an algorithm? Algorithms are said to be procedures, or recipes, so what is a procedure? What is a recipe? What are Church's and Turing's "theses"? What is "hypercomputation"?

3. What is a computer program?

What is the relation of a program to that which it models or simulates? What is simulation? Are programs (scientific) theories? Algorithms are said to be implemented in computer programs, so what is a computer program, and what is an implementation? What is software? Can computer programs be copyrighted, or patented? Can computer programs be verified?

4. What is the philosophy of artificial intelligence?

What is AI? What is the relation of computation to cognition? Can computers think? What are the Turing Test and the Chinese Room Argument?

5. What is computer ethics?

Should we trust decisions made by computers? Should we build "intelligent" computers?

PREREQUISITES:

None. But some familiarity with either computer science or philosophy would be helpful.

STAFF:**Professor:**

[Dr. William J. Rapaport](mailto:rapaport@cse.buffalo.edu), 214 [Bell Hall](#), 645-3180 x 112, rapaport@cse.buffalo.edu
Office Hours: Mondays, 11:00-11:50 a.m.; Tuesdays, 2:00-2:50 p.m.; or by appointment.

Teaching Assistant:

Dmitriy (Dima) Dligach, Trailer E-14, 645-3771, ddligach@cse.buffalo.edu
Office Hours: Tuesdays, 11:00 a.m.--1:00 p.m., or by appointment.

Note: Trailer E is between [Furnas Hall](#) & [Ketter Hall](#).

CLASS MEETINGS:

CLASS	INSTRUCTOR	REGIS. NO.	DAYS	HOURS	LOCATION
Lecture	Rapaport	CSE 410: 467274 (4 cr.) PHI 498: 306372 (3 cr.) CSE 510: 111333 (3 cr.)	TTh	9:30 - 10:50 a.m.	NSC 222

RECOMMENDED TEXTS:

There are no good texts for this course; hence, there are no *required* texts. Each of the following *recommended* texts overlaps to some extent the topics we will cover. Most of the assigned readings, however, will be made available on the Web or as class handouts.

1. [Colburn, Timothy R.](#) (2000), *Philosophy and Computer Science* (Armonk, NY: M.E. Sharpe); ISBN 1-56324-991-X.
2. [Floridi, Luciano](#) (1999), *Philosophy and Computing: An Introduction* (London: Routledge); ISBN 0-415-18025-2.
 - [Webliography](#)
3. [Floridi, Luciano](#) (2004), *The Blackwell Guide to the Philosophy of Computing and Information* (Malden, MA: Blackwell); ISBN 0-631-22919-1.
4. [Woodhouse, Mark B.](#) (2003), *A Preface to Philosophy, 7th edition* (Wadsworth Publishing); ISBN 0534595448.

IMPORTANT DATES & TENTATIVE SCHEDULE:

Note: This is the most tentative schedule I have ever created, since this is a brand-new course. I make no promises about sticking to it (not even the assignment dates, though I'll try to maintain those as much as possible)! **UPDATED** I have adjusted some of the dates below to reflect what we actually did in class, rather than on what I had hoped to do:-)

DAY	MONTH	DATE	TOPIC or ASSIGNMENT	READINGS

T	Jan	13	Introduction; What is philosophy? What is philosophy of X?	See " Reading Assignments "
Th		15	What is computer science?	See " Reading Assignments "
T		20	What is science?	See " Reading Assignments "
Th		22	Assign position paper #1 What is engineering?	See " Reading Assignments "
T		27	What is a computer? (Part I: History of Computers)	See " Reading Assignments "
Th		29	Position Paper #1 due; Peer editing session #1	See " Reading Assignments "
T	Feb	3	What is an algorithm? What is computation?	See " What is computation? "
Th		5	Position Paper #1 Revisions due; Turing machines	Turing 1936 and see " Reading Assignments "
T		10	What is a computer? (Part II: Is everything a computer?)	For next time: See " Reading Assignments "
Th		12	What is a computer/Part II (concluded) What is a procedure?	See " Reading Assignments "
T		17	Optional 2nd revision of Position Paper #1 due; Assign position paper #2 What is a procedure? (concluded)	See " Reading Assignments "
Th		19	What is hypercomputation?	See " Reading Assignments "
T		24	Peer editing session #2; Deadline for abstract of optional term paper	See " Reading Assignments "
Th		26	What is hypercomputation? (concluded)	See " Reading Assignments "
T	Mar	2	Position Paper #2 Revisions due; What is a computer program?: What is an implementation?	See " Reading Assignments "
Th		4	What is an implementation? (concluded)	See " Reading Assignments "
F		5	LAST "R" DATE	
T		9	Are computer programs theories?	See " Reading Assignments "
Th		11	Assign position paper #3 What is software?	See " Reading Assignments "
Sat		13	Spring Break begins	

T		23	Class resumes; Optional 2nd revision of Position Paper #2 due. Can computer programs be copyrighted or patented?	See " Reading Assignments "
Th		25	Peer editing session #3	See " Reading Assignments "
T		30	Can programs be verified?	See " Reading Assignments "
Th	Apr	1	Optional 2nd revision of Position Paper #3 due; Assign position paper #4 Philosophy of AI: The Turing Test	See " Reading Assignments "
T		6	The Chinese-Room Argument	See " Reading Assignments "
Th		8	Peer editing session #4	See " Reading Assignments "
T		13	Assign position paper #5 What is computer ethics? Should we trust decisions made by computers?	See " Reading Assignments "
Th		15	Position Paper #4 Revisions due Is it moral to build an artificial intelligence?	See " Reading Assignments "
T		20	Peer editing session #5	See " Reading Assignments "
Th		22	Last Class: Summary & review optional, take-home final exam handed out	
T		27	Position Paper #5 Revisions due in my office (Bell 214) or mailbox (Bell 211) by 5 p.m. Reading Day	
W		28	Reading Day	
Th	May	6	Take-Home Exam xor Term Paper due in my office (Bell 214) or mailbox (Bell 211) by 5 p.m. NEW BRING YOUR READING JOURNAL TO MY OFFICE BETWEEN 10 A.M. & NOON OR BETWEEN 1 P.M. & 5 P.M.	

READING:

"Teachers open the door, but you must enter by yourself." -- Chinese Proverb

"You can lead a horse to water, but you can't make him drink." -- American Proverb

There are a lot of topics to cover, and not nearly as many lectures as there are topics. Consequently, in lectures, I will only be able to skim the surface of the issues. But I will assign a lot of reading, which I will expect you all to do. I also think it will be useful to you if you keep a "Reading Journal": For each item you read, copy interesting quotes (or at least full references to them) and--most importantly--**your comments on them and on the issues raised** in each item you read. (For suggestions on how to do this, see the "Keep a Notebook" section of my "How to Study" guide on the Web.) I will collect these Journals at the end of the semester, and include them in the grade calculation.

There are 3 levels at which you can keep up with the reading assignments:

1. **Minimal:** Just read the assigned items.
2. **Medium:** Read at the minimal level, plus read the recommended-reading items that I will announce from time to time.
3. **Maximal:** Read at the medium level, plus read some or all of the other readings that I will suggest in lecture or post to the course website or newsgroup, and/or that are listed in the bibliographies of any of these readings.

You must include all the "minimal" readings in your Journal; you may include any "medium" or "maximal" readings, too.

For advice on how to read a both philosophy and computer science writings, see "How to Read (a Computer Science Text)".

ATTENDANCE, HOMEWORKS, NEWSGROUP, ETC.:

1. You will be expected to attend all lectures (attendance will be taken), and to complete all readings and assignments on time. There will be 5 short position papers (about 1 page each), the Reading Journal, and an optional take-home final exam XOR an optional term paper.

In some cases, the short papers will really be first drafts of "position papers" on the major topics: a "before" paper in which you will be asked to give your *present* opinions on the issues. This will be followed by class discussion ("peer-editing" sessions), after which you will write an "after" paper in which you will revise your position based on your readings, the peer-editing sessions, and the class discussions. Grades on the essays will be a function of both your *ideas* and how well you *defend* and *express* them. You will have an opportunity to revise some (but not all) of these.

2. All position papers and assignments will be announced in lecture. Therefore, be sure to get a classmate's phone number or email address (for instance, 1 or 2 people sitting next to you in class, whoever they are!) so that you will not miss assignments in the unlikely event that you miss a class.

Most, but not necessarily all, assignments will also be posted to the course website or newsgroup.

3. On the other hand, some urgent announcements will only be posted to the newsgroup! So you should subscribe to, and regularly monitor, the newsgroup sunyab.cse.510. You may also post

questions and comments there that are of general interest to the entire class. If you send me email that I deem to be of general interest, I will feel free to post it anonymously to the newsgroup along with my reply *unless you explicitly tell me otherwise*.

- Students should notify Prof. Rapaport within the first two weeks of class if they have a disability which would make it difficult to carry out course work as outlined (requiring note-takers, readers).

HOW TO STUDY:

For general advice on how to study for *any* course, see my web page, "[How to Study](#)".

GRADING:

Undergrads (in 410/498) and grads (in 510) will be graded on different bases. All graded work will receive a letter grade: 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-' (410/498 only), 'D+' (410/498 only), 'D', or 'F'. Not all work turned in will be graded; however, all work turned in will be recorded, and missing work (and missing class) will tend to lower your grade. The take-home final exam and the term paper are optional; you may not do both, however. The final letter grade will be a weighted average of all required work at either of three levels:

- Minimal:** Attend all lectures, participate in class or on-line discussions, participate in all peer-editing sessions (whether or not you choose to write the position papers), do all assigned readings, and turn in a complete Reading Journal. **The maximum grade such students can receive is C.**
- Medium:** Do all work required at the Minimal level, **plus** write all 5 position papers. **The maximum grade such students can receive is B.**
- Maximal:** Do all work required at the Medium level, **plus** write *either* the take-home exam *or else* a term paper (but not both). **The maximum grade such students can receive is A.**

Weights:

	attendance	20%
minimal:	participation (in class, on newsgroup, and/or peer-editing sessions)	10%
	reading journal	20%
medium:	= minimum + 5 position papers	25%
maximal:	= medium +: final exam xor <u>term paper</u>	25%

These weightings automatically guarantee that a student who gets "A" grades for each component of the minimal level will get a C in the course, that a student who gets an A for each component of the medium level will get a B in the course, and that a student who gets an A for each component of the maximal level will get an A in the course.

Note that even if you do all the work at any level, you might still get a grade lower than indicated above if, for instance, you did not attend all lectures or if your letter grade for the papers or exam is less than A, etc.

Note to (CSE) graduate students taking 510: In order to get a minimum passing grade for credit towards your degree, you will need to do work at least at the "medium" level of participation in order to get at least a B-.

I will post more information on both the mechanics of the position papers (and peer editing sessions) and the term paper later in the semester.

For further information on my philosophy of grading, see my web document on "Grading Principles"

Incompletes:

It is University policy that a grade of Incomplete is to be given only when a small amount of work or a single exam is missed due to circumstances beyond the student's control, and that student is otherwise doing passing work. I will follow this policy *strictly*! Thus, you should assume that I will *not* give incompletes :-)

Any incompletes that I might give, in a lapse of judgment :-), will have to be made up by the end of the *Fall 2004* semester.

For more information on Incomplete policies, see the web page, "Incompletes".

ACADEMIC INTEGRITY:

While it is acceptable to discuss general approaches with your fellow students, the work you turn in must be your own. **It is the policy of this department that any violation of academic integrity will result in an F for the course, that all departmental financial support including teaching assistanceship, research assistanceship, or scholarships be terminated, that notification of this action be placed in the student's confidential departmental record, and that the student be permanently ineligible for future departmental financial support.** If you have any problems doing the assignments, consult Prof. Rapaport. Please be sure to read the webpage, "Academic Integrity: Policies and Procedures", which spells out all the details of this, and related, policies.

CLASSROOM DISRUPTIONS:

In large classes (such as this), students have been known to be disruptive, either to the instructor or to fellow students. The university's policies on this topic, both how the instructor should respond and how students should behave, may be found in the document "Obstruction or Disruption in the Classroom - Policies" (PDF).

file: 510/syl-2004-04-20.html

UNIVERSITY AT BUFFALO - STATE UNIVERSITY OF NEW YORK
The Department of Computer Science & Engineering

cse@buffalo

CSE 4/510 & PHI 498: Phil. of Comp. Sci. Directory of Documents



- [Phil of CS Homepage](#)
- [Phil of CS Syllabus](#)
 - (the syllabus was originally handed out Jan 13, and is continually being updated; changes will be announced in class and/or in the newsgroup)
- [Go to: newsgroup](#)
- [Newsgroup Archive](#)
- [How to Study](#)
- [How to Write](#)
- [CIT Services for Students](#)
 - A list of links to services provided by [Computing and Information Technology](#).

Last Update: 12 May 2004

Note: **NEW** or **UPDATED** material is highlighted

-
- **NEW GRADES ARE SUBMITTED; PAPERWORK AVAILABLE!**
 - FINAL EXAM
 - READING ASSIGNMENTS
 - POSITION-PAPER ASSIGNMENTS
 - TERM-PAPER TOPICS
-

1. What is the philosophy of computer science?
2. What is philosophy?
3. What is computer science?
 - What is science?

- What is engineering?
 - What is a computer?
 - What is an algorithm? What is computation?
 - What is a procedure?
 - What is hypercomputation?
4. What is a computer program?
- Containing:
- What is implementation?
 - "Formal Systems"
 - Are programs theories?
 - What is Software?
- a. Can programs be copyrighted or patented?
b. Can programs be verified?
5. Philosophy of artificial intelligence
6. Computer Ethics
7. Course Summary

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/directory-2004-05-12.html

Reading Assignments

Last Update: 18 April 2004

Note: **NEW** or **UPDATED** material is highlighted

Notes:

1. Reading assignments are listed in reverse chronological order.
 2. All readings are accessible from "*buffalo.edu*" machines via the appropriate ["Directory of Documents"](#) webpages indicated below (next to "Topic").
-

1. **NEW** Assigned: 20 Apr 04
Topic: Philosophy of Computer Science

- a. Scheutz 2002
 - b. Smith 2002
-

2. Assigned: 6 Apr 04
Topic: Computer Ethics

Required:

- a. Moor 1979
 - Highly recommended!!!: Johnson 2002
 - Recommended: Friedman & Kahn 1992
- b. Lachat 1986
 - Highly recommended!!!!!!!!!!!!!!: Lem 1971

Recommended:

- Moor 1985
-

3. Assigned: 30 Mar 04
Topic: Philosophy of AI

- a. Turing 1950

b. Searle 1980

4. **Assigned: 23 Mar 04**
Topic: Can programs be verified?

Required:

- Smith 1985

Strongly recommended:

- Fetzer 1988
 - Recommended background for Fetzer 1988:
 - De Millo et al. 1979
 - Ardis et al. 1989
-

5. **Assigned: 11 Mar 04**
Topic: Can programs be copyrighted or patented?

Required:

- Newell 1985-1986

Recommended:

- Samuelson 1990
 - Koepsell & Rapaport 1995
-

6. **Assigned: 4 Mar 04**
Topic: What is software?

Required:

- Moor 1978

Strongly Recommended:

- Suber 1988

Recommended:

- Colburn 1999
-

7. **Assigned: 2 Mar 04**
Topic: What is a computer program/Are programs theories?

Required:

- a. Read the passages quoted from Johnson-Laird 1981, Pylyshyn 1984, and Johnson-Laird 1988 carefully.
- b. Weizenbaum 1976
- c. Simon 1996

Recommended:

- a. Wilks's chapter in Partridge & Wilks 1990 has a confusing, but useful, overview of the many meanings of "theory" and "model".
 - b. Daubert v. Merrell 1993 has interesting observations on the nature of scientific theories and expertise.
 - c. Green 2004 has a useful survey of different views of scientific explanation and scientific models embedded in a discussion of connectionism.
-

8. **Assigned: 24 Feb 04**

Topic: What is a computer program/What is implementation?

- a. Chalmers 1993a or 1993b
 - b. Rapaport 1999
-

9. **Assigned: 17 Feb 04**

Topic: What is hypercomputation?

- a. Copeland 2002 ("Hypercomputation")
 - b. Kugel 2002
-

10. **Assigned: 10 Feb 04**

Topic: What is a procedure?

- a. Preston 2000 (skim §1-2; **read §§3-4**; skim the rest)
 - b. Cleland 1993
-

11. **Assigned: 5 Feb 04**

Topic A: What is an algorithm?--Part II

- a. Soare 1996 or 1999, §§1-3, 4.5-5 (skim the rest)

Topic B: What is a computer?--Part II

- a. Searle 1990
 - b. Hayes 1997
-

12. **Assigned: 29 Jan 04**

Topic: What is an algorithm?--Part I

Required:

- a. Henkin 1962
 - Discusses the history of logic and the foundations of math that led up to Turing's analysis.
- b. Herman 1983
 - Discusses the informal notions of "algorithm" and "effective computability"; good background for Turing 1936.
- c. Turing 1936
 - Concentrate on the informal expository parts; the technical parts are, of course, of interest, but are rather difficult to follow, and incorrect in many parts, and can be skimmed.
 - In particular, concentrate on §§1-6 (study the simple examples of Turing machines carefully; skim the complex ones) and §9, part I (which elaborates on what it is that a *human* computer does).
 - §7 describes the universal Turing machine; §8 describes the Halting Problem. You can skim these sections.
 - If you get lost, try Suber 1997, Copeland 2004, or Copeland & Gordon's AlanTuring.net website for "gentler" expositions.

Recommended:

- d. Browse through the "Examples of Algorithms"; enjoy the cartoons :-)
 - e. Boehm & Jacopini 1966
 - f. Haugeland 1981
 - g. Soare 1996
 - Tough going in spots (you can skim those spots), but the rest is a good discussion and history of the competing analyses of "computable" (e.g., how "Turing's Thesis" is different from "Church's Thesis").
-

13. **Assigned: 27 Jan 04**

Topic: What is a computer?--Part I

Required:

- a. "A Very Brief History of Computers"; browse the linked websites.
 - b. O'Connor, J.J., & Robertson, E.F. (1998), "Charles Babbage".
 - c. Simon & Newell 1958, pp. 1-3 (on Babbage); skim the rest.
 - d. Ensmenger 2004 (soon to be on-line).
-

14. **Assigned: 20 Jan 04**

Topic: What is science?

Required (when they become available on-line):

- a. Papineau 1996: esp. pp. 290-294, 298-308, 319-320; skim the rest.
- b. Kemeny 1959: Intro., Ch.5 ("The Method"), Ch. 10 ("What Is Science?")

Recommended:

- c. Popper 1962, Ch. 1 (esp. pp. 33-59).
- d. Hempel 1966, Ch. 1.
- e. Kyburg 1968, Ch. 1 (esp. pp. 1-7).
- f. Ziman 1968.

- g. Bunge 1974.
- h. Salmon 1984, Ch. 1.
- i. Rosenberg 2000, Ch. 1.

Topic: What is engineering?

Required (when available on-line):

- a. Davis 1998: Ch. 1 (pp. 3-17), pp. 25-28, pp. 31-37; skim the rest.

Recommended:

- b. Bunge 1974, pp. 28-30; skim the rest.
 - c. Loui 1987.
 - d. Brooks 1996.
 - e. Petroski 2003.
-

15. Assigned: 15 Jan 04

Topic: What is computer science?

Read all of the following (preferably before the next lecture, since I plan on giving you *more* to read then:-), slowly (i.e., one sentence at a time) and actively (i.e., think about each sentence; make notes in your Reading Journal). Keep in mind that your main goal in reading these is to look for each author's answer to our question; consequently, at least on a first reading, you don't have to read the "irrelevant" parts quite as carefully. Below, I indicate which parts I think are of central importance for our present purposes and which parts you can just skim (i.e., read quickly):

- Newell, Perlis, & Simon 1967
- Knuth 1974, at least §1, optionally §§2-4, also.
 - (The more mathematically inclined may wish to read the whole thing :-)
- Newell & Simon 1976, pp. 113-116, 120, & "Conclusion" (pp. 125-126).
 - Skim the rest; we'll read it later. For now, concentrate just on what they have to say about what CS is.
- Denning, Comer, Gries, Mulder, Tucker, Turner, & Young 1989, pp. 9-12, 16ff; skim the rest (you can also just skim pp. 17ff).
- Hartmanis & Lin 1992, "Computer Science & Engineering" (pp. 163-168) and "Abstractions in Computer Systems" (pp. 168-174); skim the rest.
- Brooks 1996, pp. 61-64; skim the rest.
- Shapiro 2001

I also urge you to at least take a quick glance at the other items at that website.

16. Assigned: 13 Jan 04

Topic: What is philosophy?

Read at least one of the following:

- a. Woodhouse 2003, Chs. I-III (pp. 1-45).
 - A good intro to what philosophy is all about.
- b. Colburn 2000, Chs. 3-4 (pp. 19-50).

- A good survey of some of the history of philosophy that is relevant to CS.
- c. Plato, *The Apology*
- Various versions on-line
 - Plato's explanation of what Socrates thought philosophy was all about; a good intro to the skeptical, questioning nature of philosophy.
- d. Audi, Robert (2001), "Philosophy: A Brief Guide for Undergraduates" (American Philosophical Association).
- A good brief intro to what philosophy is and what its branches are.

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/readings-2004-04-06.html

What Is the Philosophy of Computer Science?

In addition to the material listed on the [syllabus](#) for this course, you might also find the following items of interest:

1. [Brown, Curtis](#) (2000), "[Seminar: Philosophy and Computers](#)"
2. [Burkholder, Leslie](#) (1992) (ed.), *Philosophy and the Computer* (Boulder, CO: Westview Press).
 - LOCKWOOD Book Collection B54 .P45 1992
3. [Bynum, Terrell Ward](#), & [Moor, James H.](#) (eds.), (2000), *The Digital Phoenix: How Computers Are Changing Philosophy, Revised Edition* (Oxford: Blackwell).
4. [Dresner, Eli](#), "Selected Topics in the Philosophy of Computer Science", at "[The Center for Digital Culture](#)"
5. [Floridi, Luciano](#), "[Philosophy and Computing: A Webliography](#)"
6. [Floridi, Luciano](#) (ed.) (2003), *The Philosophy of Information*, two special issues of *Minds and Machines* [13\(4\)](#) (free on-line!) & [14\(1\)](#).
7. [Longo, Giuseppe](#) (ed.) (1999), *Philosophy of Computer Science*, in *The Monist* [82\(1\)](#).
8. [Moor, James H.](#), & [Bynum, Terrell Ward](#) (eds.) (2002), *Cyberphilosophy: The Intersection of Computing and Philosophy* (Malden, MA: Blackwell).
9.  [Scheutz, Matthias](#) (2002), "[Philosophical Issues about Computation](#)" [PDF], *Encyclopedia of Cognitive Science* (London: Macmillan).
10. [Smith, Brian Cantwell](#) (1996), *On the Origin of Objects* (Cambridge, MA: MIT Press).
11.  [Smith, Brian Cantwell](#) (2002), "[The Foundations of Computing](#)" [PDF], in [Scheutz, Matthias](#) (ed.), *Computationalism: New Directions* (Cambridge, MA: MIT Press): 23-58.
 - For a critique of this article, see:
 - DeJohn, Jerry; & Dietrich, Eric (2003), "[Editorial: Subvert the Dominant Paradigm! A Review of Computationalism: New Directions, edited by Matthias Scheutz](#)", *Journal of Experimental & Theoretical Artificial Intelligence* [15\(4\)](#) (October-December): 375-382, esp. §3.1, pp. 378-379.
 - [direct access](#)
12. [Taylor, John](#), "[Computational Philosophy](#)"

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/whatisphilcs-2004-03-29.html

What Is Philosophy?

Last Update: 24 February 2004

Note: **NEW** or **UPDATED** material is highlighted

1. What is philosophy?

a. Plato's answer:

- "The one who feels no distaste in sampling every study, and who attacks the task of learning gladly and cannot get enough of it, we shall justly pronounce the lover of wisdom, the philosopher."

-- Plato, Republic V,475c (trans. Paul Shorey, in Edith Hamilton & Huntington Cairns (eds.) (1961), *The Collected Dialogues of Plato, including the Letters* (Princeton: Princeton University Press): 575-844, quotation on p. 714.)

b. Plato, Apology

- Provides Plato's version of Socrates's answer.

c. Audi, Robert (2001), "Philosophy: A Brief Guide for Undergraduates" (American Philosophical Association).

d. About.com's Introduction to Philosophy: Why Defining, Studying, and Doing Philosophy is Important

e. Suber, Peter (2003), "Guide to Philosophy on the Internet"

f. My 2 favorite introductions to philosophy:

i. Russell, Bertrand (1912, but there are many later editions, including one from 1997), **NEW** now online!: *The Problems of Philosophy* (various publishers).

- LOCKWOOD and UNDERGRADUATE Book Collections BD21 .R8
- LOCKWOOD Book Collection BD21 .R8 1959
- LOCKWOOD Book Collection BD21 .R8 1966

ii. Nagel, Thomas (1987), *What Does It All Mean? A Very Short Introduction to Philosophy* (Oxford: Oxford University Press).

- Asks a lot of questions; answers are left to the reader :-)

g. Two on-line philosophy encyclopedias:

i. Zalta, Edward N. (ed.), Stanford Encyclopedia of Philosophy

- ii. Fieser, James (ed.), Internet Encyclopedia of Philosophy
 - h. Two off-line philosophy encyclopedias:
 - i. Craig, Edward (ed.) (1998), Routledge Encyclopedia of Philosophy (London: Routledge)
 - Parts may be online for free at: "REP Online"
 - LOCKWOOD and UNDERGRADUATE Reference B51 .R68 1998
 - LOCKWOOD Reference CD-ROM B51 .R68 1998
 - ii. Edwards, Paul (ed.) (1967), *Encyclopedia of Philosophy* (New York: Macmillan)
 - LOCKWOOD Book Collection B41 .E5
 - 1972 reprint edition: LOCKWOOD Book Collection B41 .E5 1972
 - 1972 reprint edition: LOCKWOOD and UNDERGRADUATE Reference B41 .E5 1972
-

2. Can there be progress in philosophy? Can philosophy ever solve any of its problems?

- a. Rapaport, William J. (1982), "Unsolvable Problems and Philosophical Progress", *1982 Prize Essay, American Philosophical Quarterly* 19: 289-298.

Abstract: Philosophy has been characterized (e.g., by Benson Mates) as a field whose problems are unsolvable. This has often been taken to mean that there can be no progress in philosophy as there is in mathematics or science. The nature of problems and solutions is considered, and it is argued that solutions are always parts of theories, hence that acceptance of a solution requires commitment to a theory (as suggested by William Perry's scheme of cognitive development). Progress can be had in philosophy in the same way as in mathematics and science by knowing what commitments are needed for solutions. Similar views of Rescher and Castañeda are discussed. (See also Rapaport 1984.)

- b. Rapaport, William J. (1984), "Can Philosophy Solve Its Own Problems?" *The [SUNY] News* 13 (May/June 1984) F2-F3.

Abstract: A popularization of Rapaport 1982, discussing whether progress can be made in solving three classical philosophical problems (free will vs. determinism, skepticism, and the Liar paradox). Discusses Perry's cognitive-developmental scheme.

3. Some articles on metaphysics:

- a. Quine, Willard van Orman (1948), "On What There Is" [PDF], *Review of Metaphysics* 2(5): 21-38.
 - Reprinted in:
 - Quine, *From a Logical Point of View: 9 Logico-Philosophical Essays, Second Edition, revised* (Cambridge, MA: Harvard University Press, 1980): 1-19.
- b. A computational perspective on non-existents and other "intensional" items:

- Hirst, Graeme (1991), "Existence Assumptions in Knowledge Representation", Artificial Intelligence 49: 199-242.
-

4. On epistemology; what is knowledge?

- a. Gettier, Edmund L. (1963), "Is Justified True Belief Knowledge?", Analysis 23: 121-23.
 - Reprinted in:
A.P. Griffiths (ed.), Knowledge and Belief (Oxford: Oxford University Press, 1967).
-

5. How to study philosophy:

- a. Suber, Peter, "Courses"
 - See especially the links at the bottom of the page under the heading "Files and links that pertain to more than one course"
 - b. Wadsworth Press Philosophy Student Survival Guide
 - c. Google search on "Writing Philosophy Papers"
-

6. Miscellaneous:

- a. Dennett, Daniel (ed.), "The Philosophical Lexicon"
 - A satirical dictionary of philosophical jargon. Published by Blackwell in association with the American Philosophical Association.
 - b. McGinn, Colin (2003), "Finding Philosophy", Prospect, Issue 92 (November).
 - A brief autobiography of how a well-known contemporary philosopher got into the field.
-

What Is Computer Science?

Last Update: 28 January 2003

Note: **NEW** or **UPDATED** material is highlighted

Significant items are starred; items are listed in chronological order.

1. * Newell, Allen; Perlis, Alan J.; & Simon, Herbert A. (1967), "Computer Science", *Science* 157(3795) (22 September): 1373-1374.
2. * Knuth, Donald (1974), "Computer Science and Its Relation to Mathematics", *American Mathematical Monthly* 81(4) (April): 323-343.
3. * Newell, Allen, & Simon, Herbert A. (1976), "Computer Science as Empirical Inquiry: Symbols and Search", *Communications of the ACM* 19(3) (March): 113-126.
4. Arden, Bruce W. (1980), "COSERS Overview" [PDF], in Bruce W. Arden (ed.), *What Can Be Automated? The Computer Science and Engineering Research Study (COSERS)* (Cambridge, MA: MIT Press), Ch. 1, pp. 1-31.
 - See esp. the sections titled "About Names and Labels" and "Some Definitions", pp. 5-10.
 - See also the (untitled) preface on the (unnumbered) pages immediately preceding p. 1.
 - Contains the following "sample questions" that are "more specific" than "What can be automated?":
 - i. Are there useful, radically different computer organizations yet to be developed?
 - ii. What aspects of programming languages make them well suited for the clear representation of algorithms in specific subject areas?
 - iii. What principles govern the efficient organization of large amounts of data?
 - iv. To what extent can computers exhibit intelligent behavior?
 - v. What are the fundamental limits on the complexity of computational processes?
 - vi. What are the limits on the simulation of physical systems?
 - vii. How can processes be distributed in a communicating network to the advantage of microtechnology?
 - viii. To what extent can numerical computation be replaced by the computer manipulation of symbolic expressions?

ix. Is it possible to replace a major part of system observation and testing with automated proofs of the correctness of the constituent algorithms?

◦ Also contains the following list of "subject areas" of CS:

- i. Numerical computation
- ii. Theory of computation
- iii. Hardware systems
- iv. Artificial intelligence
- v. Programming languages
- vi. Operating systems
- vii. Database systems
- viii. Software methodology

From this, he concludes that "What can be automated?" means: "What are the limits, imposed by complexity, on the application of computers?"

5. Boorstin, Daniel J. (1983), *The Discoverers* (New York: Random House), Ch. 49: "The Microscope of Nature":

- "The hero of this story, Marcello Malpighi (1628-1694), was a great scientist whose work had no dogmatic unity. He was one of the first of a new breed of explorers who defined their mission neither by the doctrine of their master nor by the subject that they studied. They were no longer "Aristotelians" or "Galenists." Their eponym, their mechanical godparent, was some device that extended their senses and widened their vistas. What gave his researches coherence was a new instrument. Malpighi was to be a "microscopist," and his science was "microscopy," a word first noted in English in Pepys' Diary in 1664. His scientific career was held together not by what he was trying to confirm or to prove, but by the vehicle which carried him on his voyages of observation." (p. 376.)

6. Krantz, Steven G. (1984), Letter to the Editor about the relation of computer science to mathematics, *American Mathematical Monthly* 91(9) (November): 598-600.

7. **UPDATED** --correct version now online!!!

Denning, Peter J. (1985), "What Is Computer Science?" [PDF], *American Scientist* 73 (January-February): 16-19.

8. Abrahams, Paul (1987), "What Is Computer Science?", *Communications of the ACM* 30(6) (June): 472-473.

9. Loui, Michael C. (1987), "Computer Science Is an Engineering Discipline" [PDF] *Engineering Education*.

10. * Denning, Peter J.; Comer, Douglas E.; Gries, David; Mulder, Michael C.; Tucker, Allen; Turner, A. Joe; & Young, Paul R. (1989), "Computing as a Discipline", *Communications of the ACM* 32(1) (January): 9-23.

◦ "A Taxonomy of Subfields in CS&E:

- Algorithms and data structures
- Programming languages
- Computer architecture
- Numeric and symbolic computation
- Operating systems

- Software engineering
- Databases and information retrieval
- Artificial intelligence and robotics
- Human-computer interaction"

11. **UPDATED** --now online!!
Bajcsy, Ruzena K.; Borodin, Allan B.; Liskov, Barbara H.; & Ullman, Jeffrey D. (1992), "Computer Science Statewide Review" (unpublished report).
12. Hartmanis, Juris, & Lin, Herbert (eds.?) (1992), "What Is Computer Science and Engineering?" [postscript] [PDF], in Juris Hartmanis & Herbert Lin (eds.), Computing the Future: A Broader Agenda for Computer Science and Engineering (Washington, DC: National Academy Press), Ch. 6, pp. 163-216.
- NOW ON ON-LINE RESERVE; JUST CLICK ON "PDF" ABOVE
13. Abelson, Harold, & Sussman, Gerald Jay, with Sussman, Julie (1996), Structure and Interpretation of Computer Programs, "Preface to the First Edition":
- "Underlying our approach to this subject is our conviction that "computer science" is not a science and that its significance has little to do with computers. The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called procedural epistemology -- the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects. Mathematics provides a framework for dealing precisely with notions of "what is." Computation provides a framework for dealing precisely with notions of "how to." "
14. * Brooks, Frederick P., Jr. (1996), "The Computer Scientist as Toolsmith II", *Communications of the ACM* 39(3) (March): 61-68.
15. Gal-Ezer, Judith, & Harel, David (1998), "What (Else) Should CS Educators Know?", *Communications of the ACM* 41(9) (September): 77-84.
- Contains a section titled "What is CS?"
 - Contains a "Bibliography for `What is CS?'"
 - Also contains the following quotes:
 - i. "Computer science has such intimate relations with so many other subjects that it is hard to see it as a thing in itself." -- Marvin Minsky
 - ii. "Computer science differs from the known sciences so deeply that it has to be viewed as a new species among the sciences." -- Juris Hartmanis
16. Jacob, Christian (1999), "What Is Computer Science?" [PDF]
17. Shagrir, Oron (1999), "What Is Computer Science About?" [PDF], *The Monist* 82(1): 131-149.
- NOW ON ON-LINE RESERVE; JUST CLICK ON THE TITLE ABOVE!
18. Computer Science and Telecommunications Board (2001), "Fundamentals of Computer Science: Symposium" (conference program)
19. Johnson, George (2001), "All Science Is Computer Science", *The New York Times* (25 March):

WK1, WK5.

- On-line with original photos at www.ezboard.com
20. * [Shapiro, Stuart C.](#) (2001), "[Computer Science: The Study of Procedures](#)" [PDF]
 - Note: There is also an earlier, slightly different version, apparently no longer on line: Shapiro, Stuart C. (1997), "What Is Computer Science?".
 - Also see his website "[Computer Science](#)"
 21. [Foley, Jim](#) (2002), "[Computing > Computer Science](#)".
 - Revised version of paper that appeared in *Computing Research News* 14(4) (September) (2002): 6.
 22. [Roberts, Eric](#) (2002), "[What Is Computer Science?](#)"
 23. Boston University [Department of Computer Science](#) (2003), "[What Is Computer Science?](#)" [PDF]
 24. [Hammond, Tracy Anne](#), "[What Is Computer Science? Myths vs. Truths](#)"
 25. [Summary](#)

Copyright © 2004 by [William J. Rapaport](#) (rapaport@cse.buffalo.edu)
file: 510/whatiscs.2004.01.28.html

What Is Computer Science?

Summary

As we have seen, there are several different, but related, answers to the question "[What is computer science?](#)". Here is an outline summary of some of the answers.

Answers differ on:

1. **Category:**

science vs. "(systematic) study" vs. engineering

2. **Qualification (of "science"):**

(mathematical vs. empirical) vs. (natural vs. artificial)

3. **Subject matter:**

- a. computers (plus related phenomena)
- b. algorithms (plus related phenomena)
 - i. procedures (plus related phenomena)
- c. information (plus related phenomena)

- These may be [extensionally equivalent](#), depending on what the "related phenomena" are in each case.

4. **XOR:** procedural (computational?) epistemology

We'll discuss some (e.g., science vs. engineering, computers, algorithms (and procedures)), but not all (e.g., math vs. empirical vs. natural vs. artificial), of these topics later in the semester. (Things not discussed are potential [term-paper topics](#).) (Things *discussed* are potential final-exam topics, of course:-)

What Is Science?

(and What Is a Scientific Theory?)

(and Readings on Philosophy of Science)

Last Update: 2 February 2004

Note: **NEW** or **UPDATED** material is highlighted

Significant items are starred; items are listed in chronological order.

1. * Quine, Willard van Orman (1951), "Two Dogmas of Empiricism" *Philosophical Review* 60: 20-43.
2. Kemeny, John G. (1959), *A Philosopher Looks at Science* [Intro & Chs.5,10 in PDF] (Princeton: D. van Nostrand).
3. * Popper, Karl R. (1959), *The Logic of Scientific Discovery* (New York: Harper & Row).
4. * Kuhn, Thomas S. (1962) *The Structure of Scientific Revolutions* (Chicago: University of Chicago Press).
5. * Popper, Karl R. (1962), *Conjectures and Refutations: The Growth of Scientific Knowledge* (New York: Harper & Row).
6. * Hempel, Carl G. (1966), *Philosophy of Natural Science* (Englewood Cliffs, NJ: Prentice-Hall).
7. Kyburg, Henry E., Jr. (1968), *Philosophy of Science: A Formal Approach* (New York: Macmillan).
8. Ziman, John M. (1968), "What Is Science?" [PDF], from John M. Ziman, *Science Is Public Knowledge* (Cambridge, UK: Cambridge University Press).
 - Reprinted in:
 - Michalos, Alex C. (ed.) (1974), *Philosophical Problems of Science and Technology* (Boston: Allyn & Bacon): 1-27.
9. Arden, Bruce W. (1980), "COSERS Overview" [PDF] in Bruce W. Arden (ed.), *What Can Be Automated? The Computer Science and Engineering Research Study (COSERS)* (Cambridge, MA: MIT Press), Ch. 1, pp. 1-31.
 - See esp. the section titled "About Names and Labels" (pp. 5-7).
10. * Salmon, Wesley C. (1984), *Scientific Explanation and the Causal Structure of the World* (Princeton: Princeton University Press).

11. Abrahams, Paul (1987), "What Is Computer Science?", *Communications of the ACM* 30(6) (June): 472-473.
12. Denning, Peter J.; Comer, Douglas E.; Gries, David; Mulder, Michael C.; Tucker, Allen; Turner, A. Joe; & Young, Paul R. (1989), "Computing as a Discipline", *Communications of the ACM* 32(1) (January): 9-23.
13. Hartmanis, Juris, & Lin, Herbert (eds.?) (1992), "What Is Computer Science and Engineering?" [[postscript](#)] [[PDF](#)], in Juris Hartmanis & Herbert Lin (eds.), *Computing the Future: A Broader Agenda for Computer Science and Engineering* (Washington, DC: National Academy Press), Ch. 6, pp. 163-216.
 - NOW ON ON-LINE RESERVE; JUST CLICK ON "PDF" ABOVE
14. Abelson, Harold, & Sussman, Gerald Jay, with Sussman, Julie (1996), *Structure and Interpretation of Computer Programs*, "Preface to the First Edition":
 - "Underlying our approach to this subject is our conviction that "computer science" is not a science and that its significance has little to do with computers. The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called procedural epistemology -- the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects. Mathematics provides a framework for dealing precisely with notions of "what is." Computation provides a framework for dealing precisely with notions of "how to." "
15. Papineau, David (1996), "Philosophy of Science" [[PDF](#)], in Nicholas Bunnin & E.P. Tsui-James (eds.), *The Blackwell Companion to Philosophy* (Oxford: Blackwell): 290-324.
16. Simon, Herbert A. (1996), *The Sciences of the Artificial*, 3rd edition (Cambridge, MA: MIT Press).
 - SCI/ENGR Book Collection Q175 .S564 1996
 - **NEW** "Natural science is knowledge about natural objects and phenomena. We ask whether there cannot also be "artificial" science--knowledge about artificial objects and phenomena." (p. 3.)
 - **NEW** Ch. 1 ("The Natural and Artificial Worlds") has sections on the nature of understanding by simulating and on computers as artifacts, as abstract objects, and as empirical objects
17. * Rosenberg, Alex (2000), *Philosophy of Science: A Contemporary Introduction* (London: Routledge).

What Is Engineering?

And Is Computer Science an Engineering Discipline?

Last Update: 2 February 2004

Note: **NEW** or **UPDATED** material is highlighted

Significant items are starred; items are listed in chronological order.

1. Bunge, Mario (1974), "Towards a Philosophy of Technology" [PDF], in Michalos, Alex C. (ed.), *Philosophical Problems of Science and Technology* (Boston: Allyn & Bacon): 28-47.
2. Arden, Bruce W. (1980), "COSERS Overview" [PDF] in Bruce W. Arden (ed.), *What Can Be Automated? The Computer Science and Engineering Research Study (COSERS)* (Cambridge, MA: MIT Press), Ch. 1, pp. 1-31.
 - See esp. the section titled "About Names and Labels" (pp. 5-7).
3. Abrahams, Paul (1987), "What Is Computer Science?", *Communications of the ACM* 30(6) (June): 472-473.
4. * Loui, Michael C. (1987), "Computer Science Is an Engineering Discipline" *Engineering Education*.
5. Denning, Peter J.; Comer, Douglas E.; Gries, David; Mulder, Michael C.; Tucker, Allen; Turner, A. Joe; & Young, Paul R. (1989), "Computing as a Discipline", *Communications of the ACM* 32(1) (January): 9-23.
6. Hartmanis, Juris, & Lin, Herbert (eds.?) (1992), "What Is Computer Science and Engineering?" [postscript] [PDF], in Juris Hartmanis & Herbert Lin (eds.), *Computing the Future: A Broader Agenda for Computer Science and Engineering* (Washington, DC: National Academy Press), Ch. 6, pp. 163-216.
 - NOW ON ON-LINE RESERVE; JUST CLICK ON "PDF" ABOVE
7. Florman, Samuel C. (1994), *The Existential Pleasures of Engineering, 2nd edition* (New York: St. Martin's Press).
 - SCI/ENGR Book Collection TA157 .F57 1994
 - **NEW** "It is generally recognized...that engineering is "the art of science of making practical application of the knowledge of pure sciences".... The engineer uses the logic of science to achieve practical results." (pp. x-xi.)
8. Abelson, Harold, & Sussman, Gerald Jay, with Sussman, Julie (1996), *Structure and Interpretation of Computer Programs, "Preface to the First Edition"*.

9. * Brooks, Frederick P., Jr. (1996), "The Computer Scientist as Toolsmith II", *Communications of the ACM* 39(3) (March): 61-68.
10. * Davis, Michael (1998), *Thinking Like an Engineer: Studies in the Ethics of a Profession* (New York: Oxford University Press).
 - See esp. Part I ("Introduction to Engineering")
11. Davis, Martin (2000), "Introduction" [postscript], in Martin Davis, *The Universal Computer* (New York: W.W. Norton).
12. * Petroski, Henry (2003), "Early [Engineering] Education", *American Scientist* 91 (May-June): 206-209.

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/whatisengg.2004.02.02.html

What Is a Computer?

Last Update: 28 June 2004

Note: **NEW** or **UPDATED** material is highlighted

1. A Very Brief History of Computers
2. Smith, Adam (1776), passage on the division of labor, from *The Wealth of Nations*.
3. O'Connor, J.J., & Robertson, E.F. (1997), "Gaspard Clair François Marie Riche de Prony"
4. Charles Babbage websites:
 - Charles Babbage Institute
 - Lee, J.A.N. (1994), "Charles Babbage".
 - O'Connor, J.J., & Robertson, E.F. (1998), "Charles Babbage"
 - ... and many more good sites locatable by doing a Google search on "Charles Babbage" (just click on his name above).
5. Joyce, David E. (1997), "The Mathematical Problems of David Hilbert"
6. Simon, Herbert A., & Newell, Allen (1958), "Heuristic Problem Solving: The Next Advance in Operations Research", *Operations Research* 6(1) (January-February): 1-10.
 - Includes a brief history of Babbage's work.
7. Davis, Martin (1987), "Mathematical Logic and the Origin of Modern Computers" [PDF], *Studies in the History of Mathematics*

Reprinted in:
Rolf Herken (ed.), *Universal Turing Machine: A Half-Century Survey; Second Edition* (Vienna: Springer-Verlag, 1995): 135-158.
8. Searle, John R. (1990), "Is the Brain a Digital Computer?", *Proceedings and Addresses of the American Philosophical Association* 64: 21-37.
 - **NEW** An interesting follow-up:
Piccinini, Gualtiero (2003), "The Mind as Neural Software: Functionalism, Computationalism, and Computational Functionalism", paper read at the APA Pacific Division (March 2004).
9. Chalmers, David (1993), "A Computational Foundation for the Study of Cognition".
 - See esp. the section "What about computers?"

10. Robinson, J.Alan (1994), "[Logic, Computers, Turing, and von Neumann](#)" [PDF], in K. Furukawa; D. Michie; & Muggleton, S. (eds.), *Machine Intelligence 13: Machine Intelligence and Inductive Learning* (Oxford: Clarendon Press): 1-35.
 - Interesting historical comments by the developer of the resolution method of automated theorem proving on the development of computers and the related history of logic.
11. Hoyle, Michelle A. (1994-2003), "[The History of Computing Science](#)"
12. Lee, J.A.N. (1995-2002), "[The History of Computing](#)"
13. *IEEE Computer* magazine's "[Timeline of Computing History](#)" (1996)
 - This page is in html, but most of the timeline is in PDF.
14. Hayes, Patrick J. (1997), "[What Is a Computer? An Electronic Discussion](#)", *Monist* 80(3).
 - [Original emails](#)
15. Maxfield & Montrose Interactive, Inc. (1997-1998), [A History of Computers](#)
16. [Floridi, Luciano](#) (1999), *Philosophy and Computing: An Introduction* (London: Routledge), Ch. 2: "The Digital Workshop".
17. [Shagrir, Oron](#) (1999), "[What Is Computer Science About?](#)" [PDF], *The Monist* 82(1): 131-149.
18. Davis, Martin (2000), "[Introduction](#)" [postscript file] to *The Universal Computer* (New York: W.W. Norton).
19. Johnson, Mark (2002), [Review](#) of [Martin Davis's](#) *The Universal Computer: The Road from Leibniz to Turing*.
20. [Computer History Museum](#) (2003)
21. Copeland, B. Jack (2004), "Computation", in [Luciano Floridi](#) (ed.), *The Blackwell Guide to the Philosophy of Computing and Information* (Malden, MA: Blackwell).
 - Esp. pp. 3-4 on "The Birth of the Modern Computer"
 - See also: Copeland, B. Jack (2000), "[A Brief History of Computing](#)".
22. Hitmill.com (2004), [History of Computers](#)
23. Ensmenger, Nathan (2004), "[Bits of History: Review of A.R. Burks's Who Invented the Computer? The Legal Battle that Changed Computing History](#)", in *American Scientist* 91 (September-October): 467-468.
24. O'Connor, J.J., & Robertson, E.F. (2004), "[The MacTutor History of Mathematics Archive](#)"

A Very Brief History of Computers

Last Update: 2 February 2004

Note: **NEW** or **UPDATED** material is highlighted

For a more detailed history, see the Web page "[Timeline of Computing History](#)"

1832

Charles Babbage, Analytical Engine (programmable, never built);
Ada Byron Lovelace, first computer programmer

1936

Alan Turing develops what is now known as the Turing-machine model of computation.

1940

John Atanasoff & Clifford Berry: ABC electronic computer (not programmable)

1942

The Colossus computer helps the British crack German codes;
Turing works on this project

1946

John Presper Eckert & John W. Mauchly: ENIAC (Electronic Numerical Integrator and Calculator)
-- first fully electronic programmable computer

1950

UNIVAC (descendant of ENIAC) does US census;
-- first commercially marketed computer

1952

UNIVAC predicts, on live TV, that Eisenhower will win presidential election

1953

IBM 701 on sale

1975

first personal computer (Altair **UPDATED** LINK ; build-it-yourself)

1977

Apple II

1981

IBM PC

1984

Macintosh

late 1980s

Internet (network of networks)

1991

World-Wide Web

1993

first Web browser (Mosaic)

1997

50 million Web users; 15 million Internet host computers

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/history.2004.02.02.html

What Is an Algorithm?

What Is Computation?

(Starred (*) items are particularly important or interesting.)

Last Update: 29 September 2004

Note: **NEW** or **UPDATED** material is highlighted

Examples of Algorithms:

1. Some humorous examples
 2. Robertson, Jane I. (1979), "How to Do Arithmetic" *American Mathematical Monthly* 86(6) (June-July): 431-439.
 - Contains examples of algorithms for doing elementary arithmetic.
 3. Is this an algorithm?
 4. Stewart, Ian (2001), "Easter is a Quasicrystal", *Scientific American* (March): 80, 82-83.
 - Includes an algorithm for computing the date of Easter.
 5. **NEW** Phone Number Trick
-

Articles on the Nature of Algorithms

1. * Turing, Alan M. (1936), "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, Ser. 2, Vol. 42: 230-265.
 - Reprinted, with corrections, in Martin Davis (ed.), *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions* (New York: Raven Press, 1965): 116-154.
 - The on-line version above is best read using Microsoft Internet Explorer (or any other browser that uses cascaded style sheets).
 - Warning: The on-line version has several typographical errors!
 - another online version!

2. * Henkin, Leon (1962), "Are Logic and Mathematics Identical?", *Science* 138(3542) (November 16): 788-794.

- An excellent brief overview of the history of logic and the foundations of mathematics.

3. * Boehm, C., & Jacopini, G. (1966), "Flow Diagrams, Turing Machines, and Languages with only Two Formation Rules", *Communications of the ACM* 9(5): 366-371.

Abstract: This paper contains a proof that every program with gotos can be transformed into a semantically equivalent program without goto. A transformation algorithm is given.

- Discussed in:

Raskin, Jef (2003), Letter to the Editor about "Life beyond OOP", *American Scientist* 91 (May-June): 197-198.

4. Wangsness, T., & Franklin, J. (1966), "Algorithm and Formula", *Communications of the ACM* 9(4) (April): 243.

- This is a letter to the editor that is so short that I am going to reprint it in its entirety here:

Editor:

We are making this communication intentionally short to leave as much room as possible for the answers.

1. Please define "Algorithm."
2. Please define "Formula."
3. Please state the difference.

T. WANGSNESS

J. FRANKLIN

TRW Systems

Redondo Beach, California

The published answers:

- a. Huber, Harmut G.M. (1966), "Algorithm and Formula", *Communications of the ACM* 9(9) (September): 653-654.
 - b. Knuth, Donald (1966), "Algorithm and Program: Information and Data", 9(9) (September): 654.
5. Knuth, Donald (1974), "Computer Science and Its Relation to Mathematics", *American Mathematical Monthly* 81(4) (April): 323-343.
6. Weizenbaum, Joseph (1976), *Computer Power and Human Reason* (New York: W.H. Freeman).
- Ch. 2 ("Where the Power of the Computer Comes From") contains a masterful presentation of a Turing Machine implemented with pebbles and toilet paper!
 - This is also an excellent book on the role of computers in society, by the creator of the "Eliza" program.
7. Gandy, Robin (1980), "Church's Thesis and Principles for Mechanisms" [PDF], in Jon Barwise, H.J. Keisler, & K. Kunen (eds.), *The Kleene Symposium* (Amsterdam: North-Holland): 123-148.
- Argues "that Turing's analysis of computation by a human being does not apply directly to mechanical devices."
 - A follow-up article that simplifies and generalizes Gandy's paper:

- Sieg, Wilfried, & Byrnes, John (1999), "An Abstract Model for Parallel Computation: Gandy's Thesis", *The Monist* 82(1) (January): 150-164.
- Israel, David (2002), "Reflections on Gödel's and Gandy's Reflections on Turing's Thesis", *Minds and Machines* 12(2) (May): 181-201.
 - Shagrir, Oron (2002), "Effective Computation by Humans and Machines", *Minds and Machines* 12(2) (May): 221-240.
8. * Haugeland, John (1981), "Semantic Engines: An Introduction to Mind Design", in John Haugeland (ed.), *Mind Design: Philosophy, Psychology, Artificial Intelligence* (Cambridge, MA: MIT Press): 95-128.
9. * Herman, Gabor T. (1983), "Algorithms, Theory of" [PDF], in Anthony S. Ralston (ed.), *Encyclopedia of Computer Science and Engineering, 2nd edition* (New York: Van Nostrand Reinhold): 57-59.
10. Rapaport, William J. (1985), "Turing Machines" [PDF], from Morton L. Schagrin, Randall R. Dipert, & William J. Rapaport, *Logic: A Computer Approach* (New York: McGraw-Hill): 327-339.
11. Chisum, Donald S. (1985-1986), "The Patentability of Algorithms", *University of Pittsburgh Law Review* 47: 959-1022.
- See "Can Programs be Copyrighted or Patented?"
12. Davis, Martin (1987), "Mathematical Logic and the Origin of Modern Computers" [PDF], *Studies in the History of Mathematics*
- Reprinted in:
Rolf Herken (ed.), *Universal Turing Machine: A Half-Century Survey; Second Edition* (Vienna: Springer-Verlag, 1995): 135-158.
13. * Dewdney, A.K. (1989), "Algorithms: Cooking Up Programs", "Turing Machines: The Simplest Computers", and "Universal Turing Machines: Computers as Programs", Chs. 1, 28, & 48 from Dewdney's *The Turing Omnibus: 61 Excursions in Computer Science* (Rockville, MD: Computer Science Press).
- Included primarily for Ch. 48 on Universal Turing Machines, with the earlier chapters included for the sake of completeness.
14. Crossley, John N., & Henry, Alan S., "Thus Spake Al-Khwarizmi: A Translation of the Text of Cambridge University Library ms. Ii.vi.5", *Historia Mathematica* 17(2) (1990): 103-131.
- SCI/ENGR Periodical Collection Per QA21 .H54
15. Harnad, Stevan (ed.), Special Issue on "What Is Computation?", *Minds and Machines* 4(4).
- Contents:
- Harnad, Stevan, "Preface", pp. 377-378.
 - * Harnad, Stevan, "Computation Is Just Interpretable Symbol manipulation; Cognition Isn't", pp. 379-390.
 - * Chalmers, David J., "On Implementing a Computation", pp. 391-402.
 - Chrisley, Ronald L., "Why Everything Doesn't Realize Every Computation", pp. 403-420.
 - MacLennan, Bruce J., "Words Lie in Our Way", pp. 421-437.

- Kentridge, Robert W., "Symbols, Neurons, Soap-Bubbles, and the Neural Computation underlying Cognition", pp. 439-449.
 - Boyle, C. Franklin, "Computation as an Intrinsic Property", pp. 451-467.
 - * Bringsjord, Selmer, "Computation, among Other Things, Is beneath Us", pp. 489-490.
16. * Copeland, B. Jack (1996), "What Is Computation?" [PDF of preprint version], *Synthese* 108: 335-359.
17. * Soare, Robert I. (1996), "Computability and Recursion", *Bulletin of Symbolic Logic* 2(3) (September): 284-321.
- See also:
 - Soare, Robert I. (1999), "The History and Concept of Computability" [PDF], in E.R. Griffor (ed.), *Handbook of Computability Theory* (Amsterdam: Elsevier): 3-36.
 - "a revised and shortened form" of Soare 1996.
18. Suber, Peter (1997), "Turing Machines" (a 2-part handout; click on "second hand-out" at the end of part I to get to part II).
19. Farkas, David K. (1999), "The Logical and Rhetorical Construction of Procedural Discourse" [PDF], *Technical Communication* 46(1) (February): 42-54.
20. Floridi, Luciano (1999), *Philosophy and Computing: An Introduction* (London: Routledge), Ch. 2: "The Digital Workshop".
21. * Preston, Beth (2000), "Recipes and Songs: Towards a Theory of Production" [PDF].
22. * Smith, Brian Cantwell (2002), "The Foundations of Computing", in Matthias Scheutz (ed.), *Computationalism: New Directions* (Cambridge, MA: MIT Press): 23-58.
23. * Blass, Andreas; & Gurevich, Yuri (2003), "Algorithms: A Quest for Absolute Definitions" [PDF], *Bulletin of the European Association for Theoretical Computer Science (EATCS)* No. 81 (October): 195-225.
24. Copeland, B. Jack (2004), "Computation", in Luciano Floridi (ed.), *The Blackwell Guide to the Philosophy of Computing and Information* (Malden, MA: Blackwell).
25. Copeland, B. Jack, & Aston, Gordon, AlanTuring.net Catalogue of Reference Articles
- Website that contains a variety of interesting papers on various aspects of Turing's work, most written by Copeland, a well-respected contemporary philosopher, including:
 1. Copeland, Jack (2000), "The Church-Turing Thesis"
 2. Copeland, Jack (2000), "What Is a Turing Machine?"
26. Rapaport, William J. (2004), "What Is Computation?"
- lecture notes

What is a heuristic?

1. * Newell, Allen, & Simon, Herbert A. (1976), "Computer Science as Empirical Inquiry: Symbols and Search", *Communications of the ACM* 19(3) (March): 113-126.
2. * Romanycia, Marc H.J., & Pelletier, Francis Jeffry (1985), "What Is a Heuristic?" [PDF], *Computational Intelligence* 1: 47-58.
3. Korf, Richard E. (1992), "Heuristics", in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, 2nd Edition* (New York: John Wiley & Sons): 611-615.
4. Shapiro, Stuart C. (1992), "Artificial Intelligence", in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, 2nd Edition* (New York: John Wiley & Sons): 54-57.
 - Revised version appears in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold, 1993): 87-90.
5. Findler, Nicholas V. (1993), "Heuristic", in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 611-612.
6. Herman, Gabor T. (1993), "Algorithms, Theory of", in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 37-39.
7. Korfhage, Robert R. (1993), "Algorithm", in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 27-29.
8. Rapaport, William J. (1998), "How Minds Can Be Computational Systems" [PDF], *Journal of Experimental and Theoretical Artificial Intelligence* 10: 403-419.

Humorous Algorithms

Last Update: 24 March 2004

Note: **NEW** or **UPDATED** material is highlighted

1. A real recipe
 - Considered as an algorithm, what's wrong with it?
 - Could you translate it into your favorite programming language (e.g., Java, Lisp, etc.)?
2. A recipe for water
3. A simple algorithm for computing your birthdate
4. "Dilbert": a simple algorithm
5. "Motley's Crew": a simple procedure
6. "Willy 'n Ethel": a recursive algorithm (with base case = mustard)???
7. "Shoe": a named procedure
8. A real procedure that is missing an instruction
9. "Hagar the Horrible": an incomplete algorithm
10. "Nancy": an ambiguous instruction
11. "Shoe": another ambiguous instruction
12. A real-life procedure with an extremely ambiguous (because highly context-dependent) first statement
13. "Nancy": a procedure for learning to ride a bike, with an infinite loop
14. **NEW** "Baby Blues": algorithm for a snowman

1. Input the first 3 digits of your phone number (not the area code).
2. Multiply by 80.
3. Add 1.
4. Multiply by 250.
5. Add the last 4 digits of your phone number.
6. Add the last 4 digits of your phone number again.
7. Subtract 250.
8. Divide by 2.
9. Output result.

Do you recognize the output?

(Source: CuriousMath.com at "[Phone Number Trick](#)")

Is This an Algorithm?

Last Update: 27 January 2003

Note: **NEW** or **UPDATED** material is highlighted

From *The New Yorker* (28 January 1991, p. 77):

CORRECTION

Some year-end prices from the New York and American stock exchanges and over-the-counter listings in the Business section Wednesday were wrong. The incorrect prices are those that should have ended in 16ths or 32nds, such as 5/16 or 25/32. Here is how to determine the correct numbers: All incorrect listings have the number 4 or 5 in them, such as a listing of 143 or 525. But not all listings that have a 4 or 5 in them are incorrect. To determine an incorrect listing, look at the other prices listed for a particular stock. All of the numbers should be relatively close together. If a stock's high for the year is $1\frac{3}{4}$ and the low is $\frac{1}{4}$ but the last price is 47, the 47 is incorrect. To determine the correct number, ignore the appropriate 4, or the 5, that appears as the price. The number, or two numbers in some cases, behind the 4 or 5 is the numerator of the fraction in which the denominator is 16 or 32. Thus, a 47 is $\frac{7}{16}$, a 413 is $\frac{13}{16}$, a 143 is $1\frac{3}{16}$. Similarly, a 53 is $\frac{3}{32}$, a 159 is $1\frac{9}{32}$, and a 557 is $5\frac{7}{32}$.

--*Phoenix (Ariz.) Arizona Republic*

[To which *The New Yorker* responded: "Questions?"]

WHAT IS COMPUTATION?

William J. Rapaport

Department of Computer Science and Engineering,
Department of Philosophy,
and Center for Cognitive Science
State University of New York at Buffalo,
Buffalo, NY14260-2000

Last Update: 2 February 2004

Note: **NEW** or **UPDATED** material is highlighted

0. Notation: $\stackrel{\text{df}}{=}$ means: "means by definition"
 $\stackrel{\text{df}}{\sim}$ means: "roughly means by definition"

1. A **function** is a set of input-output pairs.
2. A function f is **computable** $\stackrel{\text{df}}{=}$ there is an *algorithm* that *computes* f ;
 i.e., there is an algorithm A such that for all input i , $A(i) = f(i)$
 and A specifies *how* f 's input and output are related ...
3. **UPDATED**
 ... where an **algorithm** for a problem P

(the word "algorithm" comes from the name "Al-Khuwarizmi", Arab mathematician, ca. 780-850 AD)

$\stackrel{\text{df}}{=}$ a *finite procedure* (i.e., a finite set of instructions) for solving P that is:

- (a) *unambiguous* for the computer or human who will execute it;
 i.e., all steps of the procedure must be clear and well-defined for the executor, and
- (b) *effective*;
 i.e., it must eventually *halt*,
 and it must output a *correct* solution to P .

4. Great Insights of Computer Science:

I. Boole's & Shannon's Insight.

All the information about any computable problem can be represented using only 2 nouns: 0, 1 (or any other bistable pair that can flip-flop between two easily distinguishable states, such as "on"/"off", "magnetized/de-magnetized", "high-voltage/low-voltage", etc.).

- Strictly speaking, these can be used to represent discrete things; continuous things can be approximated to any desired degree, however.
- For more information, see "Great Ideas in Computer Science": Lecture Notes #2, Lecture Notes #3.
- For a literary and philosophical view of this, see:

- [Quine, Willard van Orman](#) (1987), *Quiddities: An Intermittently Philosophical Dictionary* (Cambridge, MA: Harvard University Press), "[Universal Library](#)", pp. 223-225; **NEW** very short and definitely worth reading!

II. Turing's Insight.

Every algorithm can be expressed in a language for a computer (viz., a Turing machine) consisting of an arbitrarily long paper tape divided into squares (like a roll of toilet paper, except you never run out), with a read/write head, whose only nouns are `0' and `1', and whose only verbs (or basic instructions) are:

1. move-left-1-square
2. move-right-1-square
3. print-0-at-current-square
4. print-1-at-current-square
5. erase-current-square

For more info on this particular model of Turing machines, see:

- Schagrin, Morton L.; [Rapaport, William J.](#); & [Dipert, Randall D.](#) (1985) *Logic: A Computer Approach* (New York: McGraw-Hill), **NEW** --now online!
[Appendix B \("Turing Machines"\): 327-339 \[PDF\]](#).
- For more info on other models of Turing machines, see the "Turing machine" link above.

III. Boehm and Jacopini's Insight.

Only 3 (or maybe 4 or 5) grammar rules are needed to combine any set of basic instructions into more complex ones:

1. sequence: do this; then do that
2. selection (or choice): IF such & such is the case,

THEN do this
ELSE do that
3. repetition (or looping): WHILE such & such is the case

DO this
4. HALT (optional; depends on the programming language)
5. procedure definition: Define new complex actions by name (even more optional, but very useful)

(For more info, see "Great Ideas in Computer Science": [Lecture Notes #3](#), [Lecture Notes #4](#) .

IV. The Church-Turing Thesis.

Effective computability =df Turing-machine computability

I.e., an algorithm isdf (expressible as) a Turing-machine program.

This is a *proposed* definition. How do we know that Turing-machine computability captures the intuitive notion of effective computability?

Evidence:

- Turing's analysis of computation (NB: \Leftrightarrow the Turing Test for AI!!)
- The following formalisms are all constructively equivalent (i.e., inter-compilable):
 - Turing Machines
 - Post Machines (use tape as a queue)
 - lambda calculus (Church; cf. Lisp)
 - Markov algorithms (cf. Snobol)
 - Post productions (cf. production systems)
 - Herbrand-Gödel recursion equations (cf. Algol)
 - μ -recursive functions
 - register machines

NEW For more information on some of these formalisms, see "[Structured Programming and Recursive Functions](#)" [PDF].

- Empirical evidence: All algorithms so far translate to Turing-machines

i.e., there are no intuitively effective computable algorithms that are not Turing-machine computable

5. Are there functions that are non-computable? Yes! For more info, see: "[The Halting Problem](#)"

Copyright © 1992-2004 by [William J. Rapaport](#) (rapaport@cse.buffalo.edu)
file: computation-2004-02-02.html

What Is a Procedure?

(Starred (*) items are particularly important or interesting.)

Last Update: 20 February 2004

Note: **NEW** or **UPDATED** material is highlighted

1. [The humorous examples](#) from the ["What Is an Algorithm?" website](#)
2. * [Cleland, Carol E.](#) (1993), ["Is the Church-Turing Thesis True?"](#) [PDF], *Minds and Machines* 3(3) (August): 283-312.
 - **NEW** Here's a reply, to which Cleland 1995, below, is a rejoinder:
Horsten, Leon; & Roelants, Herman (1995), "The Church-Turing Thesis and Effective Mundane Procedures", *Minds and Machines* 5(1): 1-8.
3. [Cleland, Carol E.](#) (1995), "Effective Procedures and Computable Functions", *Minds and Machines* 5(1): 9-23.

Also see these replies:

 - Israel, David (2002), "Reflections on Gödel's and Gandy's Reflections on Turing's Thesis", *Minds and Machines* 12(2) (May): 181-201.
 - Seligman, Jeremy (2002), "The Scope of Turing's Analysis of Effective Procedures", *Minds and Machines* 12(2) (May): 203-220.
4. [Farkas, David K.](#) (1999), ["The Logical and Rhetorical Construction of Procedural Discourse"](#) [PDF], *Technical Communication* 46(1) (February): 42-54.
 - An interesting look at how to write procedures (i.e., instructions), from the point of view of a technical writer.
5. * [Preston, Beth](#) (2000), ["Recipes and Songs: Towards a Theory of Production"](#) [PDF].
6. * [Cleland, Carol E.](#) (2001), "Recipes, Algorithms, and Programs", *Minds and Machines* 11(2) (May): 219-237.
 - SCI/ENGR Periodical Collection Per Q334 .M56
7. * [Shapiro, Stuart C.](#) (2001), ["Computer Science: The Study of Procedures"](#) [PDF]
8. [Cleland, Carol E.](#) (guest ed.), [Special Issue on Effective Procedures](#), *Minds and Machines* 12(2) (May).
 - SCI/ENGR Periodical Collection Per Q334 .M56

- Click on "Special Issue" above to get full table of contents with abstracts.
- * Includes the following paper:
Cleland, Carol E. (2002), "On Effective Procedures", *Minds and Machines* 12(2) (May): 159-179.

Copyright © 2004 by [William J. Rapaport \(rapaport@cse.buffalo.edu\)](mailto:rapaport@cse.buffalo.edu)
file: 510/whatisaprocedure-2004-02-20.html

What Is Hypercomputation?

(Starred (*) items are particularly useful, important, or interesting.)

Last Update: 16 March 2004

Note: **NEW** or **UPDATED** material is highlighted

Bibliographies

1. A 1994 email exchange on interactive computing, message passing, and AI actors, with numerous references.
 2. Copeland, B. Jack (2000), "Bibliographic Guide to the Field of Hypercomputation"
 - A useful bibliography
 - Also check out Copeland's homepage for further links on hypercomputation and related topics (click on his name).
 3. Burgin, Mark, & Wegner, Peter (organizers) (2003), Special Sessions on Beyond Classical Boundaries of Computability [PDF] (Parts I,II,III, & IV),2003 Spring Western Section Meeting, American Mathematical Society.
 - Excerpts from the program, plus abstracts of some of the talks.
-

Articles

1. * Wegner, Peter (1997), "Why Interaction Is More Powerful than Algorithms", *Communications of the ACM* 40(5) (May): 80-91.
 - Many of Wegner's papers are online in various formats at his homepage (click on his name).
 - Follow-up papers by Wegner:
 - a. Wegner, Peter (1999), "Towards Empirical Computer Science", *The Monist* 82(1) (January): 58-108.
 - b. Eberbach, Eugene; & Wegner, Peter (2003), "Beyond Turing Machines" **NEW** [PDF], *Bulletin of the European Association for Theoretical Computer Science (EATCS)* No. 81 (October): 279-304.
 - This is best compared to Copeland 2002 as being a good survey, with good references.

- c. Wegner, Peter & Goldin, Dina (2003), "Computation beyond Turing Machines", *Communications of the ACM* 46(4) (April): 100-102.
2. Copeland, B. Jack, & Proudfoot, Diane (1999), "Alan Turing's Forgotten Ideas in Computer Science", *Scientific American* (April): 98-103.
 - For a reply, see:
Hodges, Andrew, "The Professors and the Brainstorms"
 - Follow-up items by, or edited by, Copeland:
 - a. Copeland, B. Jack (2002), "Accelerating Turing Machines", *Minds and Machines* 12(2) (May): 303-326.
 - b. Copeland, B. Jack (guest ed.) (2002), Special Issue on Hypercomputation, *Minds and Machines* 12(4) (November).
 - Click on title for full table of contents.
 - Contains the following two, very interesting papers:
 - a. * Copeland, B. Jack (2002), "Hypercomputation" [PDF], pp. 461-502.
 - A good overview and survey.
 - b. * Kugel, Peter (2002), "Computing Machines Can't Be Intelligent (...and Turing Said So)" [PDF], pp. 563-579.
 - This paper has almost convinced me that there just might be something to all of this "hype" about "hypercomputation".
 - For more information on the application of Putnam-Gold machines to "computational learning theory", see John Case's COLT Page
 - c. Copeland, B. Jack (guest ed.) (2003), Special Issue on Hypercomputation (continued), *Minds and Machines* 13(1) (February).
 - Click on title for full table of contents.
 3. * Milner, Robin (1993), "Elements of Interaction", *Communications of the ACM* 36(1) (January): 78-89.
 4. Schächter, Vincent (1999), "How Does Concurrency Extend the Paradigm of Computation?", *The Monist* 82(1) (January): 37-57.
 5. Bringsjord, Selmer, & Zenzen, Michael (2002), "Toward a Formal Philosophy of Hypercomputation", *Minds and Machines* 12(2) (May): 259-280.
 6. Steinhart, Eric (2002), "Logically Possible Machines", *Minds and Machines* 12(2) (May): 281-301.
 7. Cotogno, P. (2003), "Hypercomputation and the Physical Church-Turing Thesis", *British Journal for the Philosophy of Science* 54(2): 181-224.

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/whatishypercomputation-2004-03-16.html

What Is a Computer Program?

(Starred (*) items are particularly useful, important, or interesting.)

Last Update: 12 July 2004

Note: **NEW** or **UPDATED** material is highlighted

What Is Implementation?

1. * Chalmers, David J. (1993a), "A Computational Foundation for the Study of Cognition" (unpublished).
 - §2 was published (in slightly different form) as "On Implementing a Computation", *Minds and Machines* 4 (1994): 391-402.
 2. * Chalmers, David J. (1993b), "Does a Rock Implement Every Finite-State Automaton?", *Synthese* 108 (1996): 309-333.
 - This is a reply to an argument similar to Searle's (1990) argument about the wall behind me implementing Wordstar, but much more detailed, due to Hilary Putnam: Putnam, Hilary (1988), Appendix [PDF] to *Representation and Reality* (Cambridge, MA: MIT Press): 121-125.
 - "*Theorem.* Every ordinary open system is a realization of every abstract finite automaton."
 3. Suber, Peter (1997), "Formal Systems and Machines: An Isomorphism".

Also see:

 - Suber, Peter (2002), "Sample Formal System S"
 4. *? Rapaport, William J. (1999), "Implementation Is Semantic Interpretation" [PDF], *The Monist* 82(1): 109-130.
 - The online version differs from the published version in being a bit longer and going into a bit more detail.
 - This article was based on a longer chapter (which it alludes to as "in preparation"), which is available online:
Rapaport, William J. (1996), "The Nature of Implementation" [postscript], Ch. 7 of my *Understanding Understanding: Semantics, Computation, and Cognition* [postscript]; pre-printed as *Technical Report 96-26* [postscript ftp] (Buffalo: SUNY Buffalo Department of Computer Science).
-

Are Programs Theories?

1. * Weizenbaum, Joseph (1976), *Computer Power and Human Reason: From Judgment to Calculation* (New York: W.H. Freeman).
 - Ch. 5 ("Theories and Models"), pp. 132-153.
 - Ch. 6 ("Computer Models in Psychology"), pp. 154-181.
 - PDF version of the above two chapters.
2. * Moor, James H. (1978), "Three Myths of Computer Science" *British Journal for the Philosophy of Science* 29(3) (September): 213-222.
3. * Johnson-Laird, Philip N. (1981), "Mental Models in Cognitive Science", in Donald A. Norman (ed.), *Perspectives on Cognitive Science* (Norwood, NJ: Ablex), Ch. 7 (pp. 147-191):
 - "Computer programming is too useful to cognitive science to be left solely in the hands of the artificial intelligenzia [sic]. There is a well established list of advantages that programs bring to a theorist: they concentrate the mind marvelously; they transform mysticism into information processing, forcing the theorist to make intuitions explicit and to translate vague terminology into concrete proposals; they provide a secure test of the consistency of a theory and thereby allow complicated interactive components to be safely assembled; they are "working models" whose behavior can be directly compared with human performance. Yet, many research workers look on the idea of developing their theories in the form of computer programs with considerable suspicion. The reason...[i]n part...derives from the fact that any large-scale program intended to model cognition inevitably incorporates components that lack psychological plausibility.... The remedy...is *not* to abandon computer programs, but to make a clear distinction between a program and the theory that it is intended to model. For a cognitive scientist, the single most important virtue of programming should come...from the buisness of developing [the program]. Indeed, the aim should be neither to simulate human behavior...nor to exercise artificial intelligence, but to force the theorist to think again." (pp. 185-186.)
4. * Pylyshyn, Zenon W. (1984), *Computation and Cognition: Toward a Foundation for Cognitive Science* (Cambridge, MA: MIT Press), Ch. 3 ("The Relevance of Computation"), pp. 48-86, esp. the section "The Role of Computer Implementation" (pp. 74-78):
 - "[T]he...requirement--that we be able to implement [a cognitive] process in terms of an actual, running program that exhibits tokens of the behaviors in question, under the appropriate circumstances--has far-reaching consequences.

One of the clearest advantages of expressing a cognitive-process model in the form of a computer program is, it provides a remarkable intellectual prosthetic for dealing with complexity and for exploring both the entailments of a large set of proposed principles and their interactions." (p. 76.)
5. * Johnson-Laird, Philip N. (1988), *The Computer and the Mind: An Introduction to Cognitive Science* (Cambridge, MA: Harvard University Press), Ch. 3 ("Computability and Mental Processes"), pp. 37-53:
 - "[T]heories of mind should be expressed in a form that can be modelled in a computer program. A theory may fail to satisfy this criterion for several reasons: it may be radically incomplete; it may rely on a process that is not computable; it may be inconsistent, incoherent, or, like a mystical doctrine, take so much for granted that it is understood only by its adherents. These flaws are not always so obvious. Students of the mind do not always

know that they do not know what they are talking about. The surest way to find out is to try to devise a computer program that models the theory." (p. 52.)

6. Partridge, Derek; & Wilks, Yorick (eds.) (1990), *The Foundations of Artificial Intelligence: A Sourcebook* (Cambridge, UK: Cambridge University Press).

Especially the following two sections, containing the articles listed (many of which are available elsewhere, some online):

- §3 ("Levels of Theory", pp. 95-118), containing:
 - a. * Marr, David (1977), "Artificial Intelligence: A Personal View", pp. 97-107; reprinted from *Artificial Intelligence* 9: 37-48.
 - b. * Boden, Margaret A., "Has AI Helped Psychology?", pp. 108-111.
 - c. Partridge, Derek, "What's in an AI Program?", pp. 112-118.
- §4 ("Programs and Theories", pp. 119-164), containing:
 - a. *Wilks, Yorick, "One Small Head: Models and Theories", pp. 121-134.
 - b. Bundy, Alan; & Ohlsson, Stellan (198?), "The Nature of AI Principles", pp. 135-154; reprinted from *AISB Quarterly* ##47-50.
 - c. Simon, Thomas W., "Artificial Methodology Meets Philosophy", pp. 155-164.

7. Daubert v. Merrell Dow Pharmaceuticals (92-102), 509 U.S. 579 (1993).

- A Supreme Court case concerning what counts as "generally accepted" reliability by the scientific community. Prof. Srihari (personal communication) wonders if certain computer programs would pass muster.

8. * Simon, Herbert A. (1996), *The Sciences of the Artificial, Third Edition* (Cambridge, MA: MIT Press), Ch. 1 ("Understanding the Natural and Artificial Worlds"), pp. 1-24 [PDF].

- This is a *really* good chapter that I probably should have had you read back when we were discussing whether computer science is a science. So read it now; it's never too late :-)

9. Scheutz, Matthias; & Peschl, Markus (2000), "Some Thoughts on Computation and Simulation in Cognitive Science" [PDF], in *Proceedings of the 6th Congress of the Austrian Philosophical Society*: 534-540.

10. Coward, L. Andrew; & Sun, Ron (2001??), "Some Criteria for an Effective Scientific Theory of Consciousness and Examples of Preliminary Attempts at Such a Theory" [PDF].

11. Peschl, Markus F.; & Scheutz, Matthias (2001), "Explicating the Epistemological Role of Simulation in the Development of Theories of Cognition" [PDF], in *Proceedings of the 7th Colloquium on Cognitive Science (ICCS-01)*: 274-280.

12. Lane, Peter C.R.; & Gobet, Fernand (2003), "Developing Reproducible and Comprehensible Computational Models" [PDF], *Artificial Intelligence* 144: 251-263.

13. Green, Christopher D. (2004), "(How) Do Connectionist Networks Model Cognition?" (forthcoming and unpublished Ph.D. dissertation, Dept. of Philosophy, University of Toronto),

Ch. 3 ("Philosophical Approaches to Explanation and Scientific Models, and Their Relations to Connectionist Cognitive Science") [.doc].

What Is Software?

1. * Moor, James H. (1978), "Three Myths of Computer Science" *British Journal for the Philosophy of Science* 29(3) (September): 213-222.
 - Moor discusses different "levels" from which to understand things like computers and programs; this notion is reminiscent of Dennett's 1971 theory of "stances": Dennett, Daniel C. (1971), "Intentional Systems", *Journal of Philosophy* 68: 87-106; reprinted in Daniel C. Dennett, *Brainstorms* (Montgomery, VT: Bradford Books): 3-22.
 - For some interesting followups to Dennett's paper, see:
Miller, Christopher A. (guest ed.) (2004), "Human-Computer Etiquette: Managing Expectations with Intentional Agents", *Communications of the ACM* 47(4) (April): 31-34.
2. * Suber, Peter (1988), "What Is Software?", *Journal of Speculative Philosophy* 2(2): 89-119.
3. * Colburn, Timothy R. (1999), "Software, Abstraction, and Ontology" [PDF], *The Monist* 82(1): 3-19.
 - Reprinted (in slightly different form) as:
Colburn, Timothy R. (2000), *Philosophy and Computer Science* (Armonk, NY: M.E. Sharpe), Ch. 12 ("Software, Abstraction, and Ontology"), pp. 198-209.

Formal Systems

Last Update: 4 March 2005

Note: **NEW** or **UPDATED** material is highlighted

Formal Systems

A **formal system** (also known as: "symbol system", "formal symbol system", "formal language", "formal theory", etc.) consists of:

1. primitive ("atomic") symbols
 - Some people understand the term "symbol" to mean a "marker" or "token" *that has a meaning*. Although I don't use that term in this way, everything I say here can be understood with either interpretation.
2. recursive rules for forming new (complex, or "molecular") symbols, ultimately from the primitive ones.
 - These are usually called "well-formed formulas", or "wffs".
3. a distinguished subset of the wffs
 - In an "axiomatic" system, these are the axioms. Although axioms are often understood as "self-evidently true" statements, I think it is better to consider them merely as wffs that are "given" initially.
4. recursive rules for forming "new" wffs from "old" ones, ultimately from the axioms
 - These rules are sometimes called "transformation rules" or--more often--"rules of inference".
 - The "new" wffs are new only in the sense that they are produced (or "generated", or "proved") from "previous" ones. Since they are wffs, they are formed (or constructed) from the primitive symbols, and so aren't "new" in the sense of never previously existing. The "old" wffs from which these new ones are generated are merely ones that are generated *before* the "new" ones.
 - A sequence of wffs (usually beginning with axioms) that is such that each wff in the sequence is either an axiom or is generated by (or "follows from") previous wffs in the sequence according to one of the rules of inference is sometimes called a "proof" (or, a "proof of" the last wff in the sequence). The last wff in the sequence is often called a "theorem".

As an analogy, consider a "Transformer" toy made of Lego blocks. For those of you who don't have (or who weren't themselves) children (probably boys) of a certain elementary-school age, Transformers are

toys that, in one configuration, look like monsters, but can be manipulated so that they turn into something else, often (believe it or not) a truck or other vehicle. Lego blocks are small building blocks that can be attached to each other to form larger structures.

1. The Lego blocks can be likened to the primitive symbols.
2. The instructions for building larger structures from the individual Lego blocks can be likened to the rules for producing wffs from primitive symbols. To continue the analogy, suppose that the only wffs are Transformers, either monsters or trucks. (This is a bit imaginary: You can't make Transformers out of Lego blocks; even if you could, the manipulations to turn them from monsters into trucks would probably cause them to fall apart.)
3. The instructions for transforming a monster into a truck, or vice versa, can be likened to the rules of inference, and if a given monster is likened to an axiom, then the truck that it can be transformed into can be likened to a theorem.

The above analogy falls apart if you look at it too closely, so please don't!

Semiotics

Semiotics (or the theory of signs and symbols) consists of:

1. Syntax

- a. Syntax is the study of the relations among the symbols of a formal symbol system. It can be more generally understood as the study of the relations among the entities of some domain, preferably a domain that can be understood more or less in the terms of a formal system.
- b. Grammatical syntax (or just "grammar", for short) is the study of which sequences (or "strings") of symbols are well formed (according to the recursive rules of grammar).
- c. Proof-theoretical syntax (or "proof theory") is the study of which sequences of wffs are "derivable" from the transformation rules (or "provable via the rules of inference").
- d. Note that my description of a formal system, above, was entirely in terms of syntax.
- e. Syntax does *not* include the study of such things as "truth", "meaning", "reference", etc.

2. Semantics

- a. Semantics is the study of the relations between the symbols of a formal system and what they represent, mean, refer to (etc.) in the "world" (where this "world" could be the real world, some possible world, some fictional world, some situation or state of affairs that is part of a world, etc.)
- b. Semantics requires:
 - i. a syntactic domain (usually a formal system); call it SYN.
 - ii. a semantic domain, characterized by an "ontology"; call it SEM.

- The ontology can be understood as a (syntactic) theory of the *semantic* domain, in the sense that it specifies:
 - the parts of the semantic domain
 - their relationships (structural [parts and wholes] and inferential [model theory, or a theory of truth and satisfaction])
 - in this sense, SEM is a "model" of SYN!
- iii. a "compositional" (i.e., homomorphic, or structure-preserving) semantic interpretation mapping from SYN --> SEM

3. Pragmatics

- a. "Pragmatics" is a grab-bag term used by different people to mean different things. The most accurate meaning is probably that pragmatics is the study of everything else that is interesting about formal systems and their interpretations that isn't covered by either syntax or semantics. Most people would agree that pragmatics includes the study of the relations among symbol systems, their interpretations, and the cognitive agents who use them. It is also often characterized as the study of "contexts" in which symbols are used. "Context", however, is another grab-bag term; it can mean "social" context, or "situational" context, or "indexical" context (e.g., the fact that the sentence "I am hungry" (which contains an "indexical" or "deictic" term: "I") means the same thing (in one sense of "means") no matter who says it (i.e., it means that the speaker is hungry), yet means different things depending on who says it (if you say it, it means that *you* are hungry, whereas if I say it, it means that *I* am hungry, which might have different truth values).

Examples

1. Hofstadter, Douglas R. (1979), *Gödel, Escher, Bach: An Eternal Golden Braid* (New York: Basic Books).
 - Ch. I ("The MU-puzzle"), pp. 33-41 [PDF].
 - MU-puzzle solution [PDF]
2. Rapaport, William J. (1996), *Understanding Understanding: Semantics, Computation, and Cognition* [postscript], pre-printed as *Technical Report 96-26* [postscript ftp] (Buffalo: SUNY Buffalo Department of Computer Science).
 - Ch. 2, §"Tarskian Semantics"
3. Suber, Peter (2002), "Sample Formal System S"

Other References:

- Haugeland, John (1981), "Semantic Engines: An Introduction to Mind Design", in John Haugeland (ed.), *Mind Design: Philosophy, Psychology, Artificial Intelligence* (Cambridge, MA: MIT Press): 95-128.
- **NEW** Posner, Roland (1992), "Origins and Development of Contemporary Syntactics",

Languages of Design 1: 37-50.

NEW

- PDF
- Tomalin, Marcus (2002), "The Formal Origins of Syntactic Theory" [PDF], *Lingua* 112(10): 827-848.
 - An interesting history of the development of formal systems and their application in contemporary linguistic theory (e.g., Chomsky).

Copyright © 2004-2005 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/formalsystems-2005-03-04.html

Can Programs Be Copyrighted or Patented?

(Starred (*) items are particularly useful, important, or interesting.)

Last Update: 22 March 2004

Note: **NEW** or **UPDATED** material is highlighted

1. * Mooers, Calvin N. (1975), "Computer Software and Copyright", *Computing Surveys* 7(1) (March): 45-72.

- o **NEW** Mooers on the distinction between an uncopyrightable **idea** and a copyrightable **expression of an idea**:

"Where does the "expression" leave off, and the "idea" take over? The best insight into this matter comes from discussions of copyright as applied to novels and dramatic productions. In these, "expression" is considered to include the choice of incident, the personalities and development of character, the choice of names, the elaboration of the plot, the choice of locale, and the many other minor details and gimmicks used to build the story. In other words, "expression" is considered to include not only the marks, words, sentences, and so on in the work, but also all these other details or structures as they aggregate into larger and larger units to make up the expression of the entire story.

"In other words, after the bare abstract "idea" has been chosen (e.g., boy meets girl, boy loses girl, boy wins girl), the "expression" to which copyright applies covers the remaining elements of original choice and artistry which are supplied by the author in order for him to develop, express, and convey his version of the bare idea." (p. 50.)

2. * Bender, David (1985-1986), "Protection of Computer Programs: The Copyright/Trade Secret Interface", *University of Pittsburgh Law Review* 47: 907-958.

3. Chisum, Donald S. (1985-1986), "The Patentability of Algorithms", *University of Pittsburgh Law Review* 47: 959-1022.

- o **NEW** Chisum, in "The Definition of an Algorithm" (Sect. B, pp. 974-977), cites several other published definitions:
 - "A method of solution for problem *P* on device [including a human being] *M* is a description in a language comprehensible to *M* of discrete steps performable by *M* and an ordering of these steps, such that given proper data, if *M* performs the prescribed steps in the prescribed order, a solution to the problem *P* will result, if one exists. A method of solution will be called a *semi-algorithm* for *P* on *M* if the solution to *P* (if one exists) appears after the performance of finitely many steps. A semi-algorithm will be called an *algorithm* if, in addition, whenever the problem has no solution the method enables the device to determine this after a finite number of steps and halts."

(R. Korfhage (1966), *Logic and Algorithms with Applications to the Computer and Information Sciences*, p. 89.)

- "...five "important features" of an algorithm:
 1. "Finiteness. An algorithm must always terminate after a finite number of steps....
 2. "Definiteness. Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case....
 3. "Input. An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins. These inputs are taken from specified sets of objects....
 4. "Output. An algorithm has one or more outputs, i.e., quantities which have a specified relation to the inputs....
 5. "Effectiveness. An algorithm is also generally expected to be *effective*. This means that all of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time by a man using pencil and paper."

(D. Knuth (1973), *The Art of Computer Programming: Fundamental Algorithms I* (2nd ed.), pp. 4-6.)

- "...three "empirical properties" that have been found to be present "in all algorithms constructed so far."
 - a. "*Determinacy*. The procedure is specified so clearly and precisely that there is no room for arbitrary interpretation. A procedure of this kind can be communicated to another person by a finite number of instructions. The operations described by these instructions do not depend on the whim of the operator and constitute a determinate process which is completely independent of the person carrying it out.
 - b. "*Generality*. An algorithm is applicable to more than just one specific problem: it is used for solving a class of problems, with the procedural instructions valid for any particular set of initial data.
 - c. "*Efficacy*. This property, sometimes called the *directionality* of an algorithm, means that application of an algorithmic procedure to any problem of a given kind will lead to a "stop" instruction in a finite number of steps, at which point one must be able to find the required solution."

(M. Aiserman et al. (1971), *Logic, Automata, and Algorithms*, pp. 308-309.)

[Chisum continues:] "Many algorithms appear to have another feature--recursiveness: one or more of the steps entails going back and repeating one or more of the prior steps." (p. 976). [!]

4. ** Newell, Allen (1985-1986), "Response: The Models Are Broken, the Models Are Broken", *University of Pittsburgh Law Review* 47: 1023-1031.

- This article should be one click away if you are on a .buffalo.edu machine on campus. If you are not, click [here](#).
- **NEW** "I think fixing the models is an important intellectual task. It will be difficult. The concepts that are being jumbled together--methods, processes, mental steps, abstraction, algorithms, procedures, determinism--ramify throughout the social and economic fabric....The task is to get...new models. There is a fertile field to be plowed here, to understand what models might work for the law. It is a job for lawyers and, importantly, theoretical computer scientists. It could also use some philosophers of computation, if we

could ever grow some." (p. 1035.)

- For follow-up (including links to commentary by Donald Knuth), see:

"Newell 1986: The Models Are Broken, The Models Are Broken!"

5. Samuelson, Pamela (1989), "Why the Look and Feel of Software User Interfaces Should Not Be Protected by Copyright Law", *Communications of the ACM* 32(5) (May): 563-572.
6. * Samuelson, Pamela (1990), "Should Program Algorithms Be Patented?", *Communications of the ACM* 33(8) (August): 23-27.
 - direct access
7. Samuelson, Pamela (1991), "Digital Media and the Law", *Communications of the ACM* 34(10) (October): 23-28.
8. Forester, Tom; & Morrison, Perry (1994), *Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computing; Second Edition* (Cambridge, MA: MIT Press).
 - esp. pp. 57-68 from Ch. 3
9. * Koepsell, David R. (2000), *The Ontology of Cyberspace: Philosophy, Law, and the Future of Intellectual Property* (Chicago: Open Court).

- **NEW** In contrast to Newell (above), Koepsell argues that it is the **law**, not computer science, that will have to change.
- Koepsell, David R.; & Rapaport, William J. (1995), "The Ontology of Cyberspace: Questions and Comments", *Technical Report 95-25* (Buffalo: SUNY Buffalo Department of Computer Science) and *Technical Report 95-09* (Buffalo: SUNY Buffalo Center for Cognitive Science).

Abstract: This document consists of two papers: "The Ontology of Cyberspace: Preliminary Questions", by David R. Koepsell, and "Comments on 'The Ontology of Cyberspace'", by William J. Rapaport. They were originally presented at the Tri-State Philosophical Association Meeting, St. Bonaventure University, 22 April 1995.

10. Johnson, Deborah G. (2001), *Computer Ethics* (Upper Saddle River, NJ: Prentice Hall).
 - Ch. 6 ("Property Rights in Computer Software"), pp. 137-167.
11. Samuelson, Pamela (2003), "Unsolicited Communications as Trespass?" [PDF], *Communications of the ACM* 46(10) (October): 15-20.

Abstract: "Attempting to stretch existing laws to address previously unforeseen technological issues."

Can Programs Be Verified?

Last Update: 30 June 2004

Note: **NEW** or **UPDATED** material is highlighted

"I hold the opinion that the construction of computer programs is a mathematical activity like the solution of differential equations, that programs can be derived from their specifications through mathematical insight, calculation, and proof, using algebraic laws as simple and elegant as those of elementary arithmetic."

"Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning."

"When the correctness of a program, its compiler, and the hardware of the computer have all been established with mathematical certainty, it will be possible to place great reliance on the results of the program, and predict their properties with a confidence limited only by the reliability of the electronics."--C.A.R. Hoare

1. * McCarthy, John (1963), "Towards a Mathematical Science of Computation", in C.M. Popplewell (ed.), *Information Processing 1962: Proceedings of DFIP Congress 62* (Amsterdam: North-Holland): 21-28; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 35-56.
2. * Naur, Peter (1966), "Proof of Algorithms by General Snapshots", *BIT* 6: 310-316; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 57-64.
3. * Floyd, Robert W. (1967), "Assigning Meanings to Programs", in *Mathematical Aspects of Computer Science: Proceedings of Symposia in Applied Mathematics, Vol. 19* (American Mathematical Society): 19-32; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 65-81.
4. ** Hoare, C.A.R. (1969), "An Axiomatic Basis for Computer Programming", *Communications of the ACM* 12: 576-580, 583; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 83-96.
5. Dijkstra, Edsgar W. (1974), "Programming as a Discipline of Mathematical Nature", *American Mathematical Monthly* (June-July): 608-612.

6. * Dijkstra, Edsger W. (1975), "Guarded Commands, Nondeterminacy and Formal Derivation of Programs", *Communications of the ACM* 18(8): 453-457
7. * De Millo, Richard A.; Lipton, Richard J.; & Perlis, Alan J. (1979), "Social Processes and Proofs of Theorems and Programs", *Communications of the ACM* 22(5): 271-280; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 297-319.
 - direct access
 - For an interesting contrasting view of the relationship between computer programs and mathematical proofs, see:

Suber, Peter (1997), "Formal Systems and Machines: An Isomorphism".
8. Naur, Peter (1982), "Formalization in Program Development", *BIT* 22: 437-453; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 191-210.
9. Dijkstra, Edsger W. (1983), "Fruits of Misunderstanding" (EWD-854), reprinted in *Datamation* (15 February 1985): 86-87.
10. Scherlis, William L.; & Scott, Dana S. (1983), "First Steps towards Inferential Programming", in R.E.A. Mason (ed.), *Information Processing 83* (JFIP and Elsevier Science Publishers): 199-212; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 99-133.
11. Olson, Steve (1984, January/February), "Sage of Software", *Science*: 75-80.
12. * Gries, David (1981), *The Science of Programming* (New York: Springer-Verlag).
13. Meyer, Bertrand (1985), "On Formalism in Specifications", *IEEE Software* 2(1) (January): 6-26; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 155-189.
14. * Smith, Brian Cantwell (1985), "Limits of Correctness in Computers" [PDF], *Technical Report CSLI-85-36* (Stanford, CA: Center for the Study of Language and Information); first published in Charles Dunlop & Rob Kling (eds.), *Computerization and Controversy* (San Diego: Academic Press, 1991): 632-646; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 275-293.
 - Also see:

Fetzer, James H. (1999), "The Role of Models in Computer Science" [PDF], *The Monist* 82(1): 20-36.
 - **NEW** Also possibly of relevance:

Jackson, Michael (2003), "Why Software Writing Is Difficult and Will Remain So" [PDF], *Information Processing Letters* 88: 13-25.
15. Verity, John W. (1985), "Bridging the Software Gap", *Datamation* (15 February): 84, 88.

16. Hoare, C.A.R. (1986), "Mathematics of Programming", *Byte* (August); reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 135-154.
17. Floyd, Christiane (1987), "Outline of a Paradigm Change in Software Engineering", in G. Bjerknes et al. (eds.), *Computers and Democracy: A Scandinavian Challenge* (Brookfield, VT: Gower Publishing): 191-210; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 239-259.
18. * Fetzer, James H. (1988), "Program Verification: The Very Idea", *Communications of the ACM* 31(9) (September): 1048-1063; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 321-358.

- direct access

1. Ardis, Mark; Basili, Victor; Gerhart, Susan; Good, Donald; Gries, David; Kemmerer, Richard; Leveson, Nancy; Musser, David; Neumann, Peter; & von Henke, Friedrich (1989), "Editorial Process Verification" (letter to the editor, with replies by James H. Fetzer and Peter J. Denning), *ACM Forum, Communications of the ACM* 32(3) (March): 287-290.

- direct access

2. Pleasant, James C.; AND Paulson, Lawrence; Cohen, Avra; & Gordon, Michael; AND Bevier, William R.; Smith, Michael K.; & Young, William D.; AND Clune, Thomas R.; AND Savitzky, Stephen (1989), "The Very Idea" (5 letters to the editor), *Technical Correspondence, Communications of the ACM* 32(3) (March): 374-377.
3. Fetzer, James H. (1989), "Program Verification Reprise: The Author's Response" (to the above 5 letters), *Technical Correspondence, Communications of the ACM* 32(3) (March): 377-381.
4. Dobson, John; & Randell, Brian (1989), "Program Verification: Public Image and Private Reality", *Communications of the ACM* 32(4) (April): 420-422.
5. Müller, Harald M.; AND Holt, Christopher M.; AND Watters, Aaron (1989), "More on the Very Idea" (3 letters to the editor, with reply by James H. Fetzer), *Technical Correspondence, Communications of the ACM* 32(4) (April): 506-512.
6. Hill, Richard; AND Conte, Paul T.; AND Parsons, Thomas W.; AND Nelson, David A. (1989), "More on Verification" (4 letters to the editor), *ACM Forum, Communications of the ACM* 32(7) (July): 790-792.
7. Tompkins, Howard E. (1989), "Verifying Feature-Bugs" (letter to the editor), *Technical Correspondence, Communications of the ACM* 32: 1130-1131.

- Note: This does not seem to be available on-line.

19. Barwise, Jon (1989), "Mathematical Proofs of Computer System Correctness" [PDF], *Notices of the American Mathematical Society* 36: 844-851.

Included in the online version of the above:

- Dudley, Richard (1990), "Program Verification" (letter to Jon Barwise (ed.), *Computers*

and Mathematics column, with a reply by Barwise), *Notices of the American Mathematical Society* 37: 123-124.

20. Blum, Bruce I. (1989), "Formalism and Prototyping in the Software Process", *Information and Decision Technologies* 15: 327-341; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 213-238.
21. Cohen, Avra (1989), "The Notion of Proof in Hardware Verification", *Journal of Automated Reasoning* 5: 127-139; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 359-374.
22. Colburn, Timothy R. (1991), "Program Verification, Defeasible Reasoning, and Two Views of Computer Science", *Minds and Machines* 1: 97-116; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 375-399.
23. Fetzer, James H. (1991), "Philosophical Aspects of Program Verification", *Minds and Machines* 1: 197-216; reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 403-427.
24. Devlin, Keith (1992), "Computers and Mathematics", *Notices of the American Mathematical Society* 39: 1065-1066.
25. MacKenzie, Donald (1992), "Computers, Formal Proofs, and the Law Courts", *Notices of the American Mathematical Society* 39: 1066-1069.
26. Naur, Peter (1992), "The Place of Strictly Defined Notation in Human Insight", in *Computing: A Human Activity* (Addison-Wesley); reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 261-274.
27. Nelson, David A. (1992), "Deductive Program Verification (A Practitioner's Commentary)", *Minds and Machines* 2: 283-307.
28. Colburn, Timothy R. (1993), "Computer Science and Philosophy", in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers): 3-31.
29. Colburn, Timothy R.; Fetzer, James H.; & Rankin, Terry L. (eds.) (1993), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers).
 - contains a lengthy bibliography
30. Nelson, David A. (1993), Review of Boyer & Moore's *A Computational Logic Handbook* and Moore's Special Issue on System Verification (*Journal of Automated Reasoning*), in *Minds and Machines* 4: 93-101.
31. Fetzer, James H. (1993), "Program Verification", in Allen Kent & James G. Williams (eds.), *Encyclopedia of Computer Science and Technology*, Vol. 28, Supp. 13 (New York: Marcel Dekker): 237-254; reprinted in Allen Kent & James G. Williams (eds.), *Encyclopedia of Microcomputers*, Vol. 14: Productivity and Software Maintenance: A Managerial Perspective to Relative Addressing (New York: Marcel Dekker): 47-64.

32. Bowen, J.P., & Hinchey, M.G. (1995), "Ten Commandments of Formal Methods", *IEEE Computer* 28(4): 56-63.
33. Glanz, James (1995), "Mathematical Logic Flushes Out the Bugs in Chip Designs", *Science* 267: 332-333.
34. Fetzer, James H. (1995), "Philosophy and Computer Science: Reflections on the Program Verification Debate", paper presented at the American Philosophical Association Central Division meeting, Chicago (April 1995).
35. Fetzer, James H. (1996), "Computer Reliability and Public Policy: Limits of Knowledge of Computer-Based Systems", *Social Philosophy and Policy*, forthcoming.
36. Neumann, Peter (1996, July), "Using Formal Methods to Reduce Risks", *Communications of the ACM* 39(7): 114.
37. Pollack, Andrew (1999, May 3), "For Coders, a Code of Conduct: 2000 Problem Tests Professionalism of Programmers", *The New York Times*: C1, C12.

Copyright © 2004 by [William J. Rapaport \(rapaport@cse.buffalo.edu\)](mailto:rapaport@cse.buffalo.edu)
file: 510/canprogsbeverified-2004-06-30.html

Philosophy of Artificial Intelligence

Last Update: 28 June 2004

Note: **NEW** or **UPDATED** material is highlighted

Entire courses have been devoted to this topic. (At UB, Prof. Randall R. Dipert (Philosophy) has taught PHI 398 Philosophy of Artificial Intelligence.)

This page only lists a few items that are relevant to our course.

Websites

1. AAAI's AI Topics website on Philosophy
 - A great site, with many links.
 2. Some Standard Sources of Information on AI
 3. Definitions of AI:
 - Two Contrasting Definitions of AI
 - Some Definitions of "Artificial Intelligence"
 4. Rapaport's freshman-level lecture notes on AI
 5. Information on the Turing Test
-

On the Mind-Body (or Mind-Brain) Problem

1. Mind-Body Theories (cartoon)
from: Taylor, Richard (1974), *Metaphysics, 2nd edition* (Englewood Cliffs, NJ: Prentice Hall): 19.
2. Bechtel, William (1988), *Philosophy of Mind: An Overview for Cognitive Science* (Hillsdale, NJ: Lawrence Erlbaum Associates).
 - LOCKWOOD Book Collection B105.M55 B43 1988
3. Colburn, Timothy R. (2000), *Philosophy and Computer Science* (Armonk, NY: M.E. Sharpe):
 - Ch. 3 ("AI and the History of Philosophy"), pp. 19-40.
 - Ch. 4 ("AI and the Rise of Contemporary Science and Philosophy"), pp. 41-50.

4. On Functionalism

- a. * Block, Ned (1996), "[What Is] Functionalism[?]", *Encyclopedia of Philosophy Supplement*
- b. ** Fodor, Jerry A. (1981, January), "The Mind-Body Problem", *Scientific American* 244(1): 114-123.
- c. Thagard, Paul (1986), "Parallel Computation and the Mind-Body Problem", *Cognitive Science* 10: 301-318.
 - An argument for the importance of the implementing medium in functionalistic theories of mind.
- d. Hilary Putnam's invention and subsequent refutation of functionalism:
 - Putnam, Hilary (1960), "Minds and Machines" [PDF], in Sidney Hook (ed.), *Dimensions of Mind: A Symposium* (New York: New York University Press): 148-179.
 - Putnam's first article on functionalism.
 - * Putnam, Hilary (1988), *Representation and Reality* (Cambridge, MA: MIT Press).
 - An argument *against* functionalism, by the philosopher who first proposed it.
- e. **NEW** Piccinini, Gualtiero (2003), "The Mind as Neural Software: Functionalism, Computationalism, and Computational Functionalism", paper read at the APA Pacific Division (March 2004).

5. On Emergent Properties

- O'Connor, Timothy; & Wong, Hong Yu (2002), "Emergent Properties", in Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2002 Edition).

Other Readings

1. * Turing, Alan M. (1950), "Computing Machinery and Intelligence", *Mind* 59: 433-460.
 - HTML version
 - another HTML version
 - For more information on the Turing Test, see the link above titled "Information on the Turing Test".
2. Fletcher, Joseph (1972), "Indicators of Humanhood: A Tentative Profile of Man", *Hastings Center Report* 2(5): 1-4.
 - On "neo-cortical function" (discussed in the above article), see: Cardoso, Silvia Helena (1997), "Specialized Functions of the Cerebral Cortex"
3. * Newell, Allen, & Simon, Herbert A. (1976), "Computer Science as Empirical Inquiry: Symbols and Search", *Communications of the ACM* 19(3) (March): 113-126.
4. * Searle, John R. (1980), "Minds, Brains, and Programs", *Behavioral and Brain Sciences* 3: 417-457.
 - The online version is the pre-print version of the "target article". BBS's policy is to have

each target article followed by about 2 dozen commentaries and a reply by the author. For the full treatment of Searle's paper, you'll need to go to the library:
 SCI/ENGR Periodical Collection Per QP360 .B425
 LOCKWOOD Periodical Collection Per QP360 .B425

- The secondary literature, both by Searle and by others, on the CRA is immense. The best brief overview is:

[Hauser, Larry](#) (2001), "[The Chinese Room Argument](#)", *Internet Encyclopedia of Philosophy*.

5. [Rapaport's webpage on how to pass a Turing test and escape from the Chinese room](#)
6. [Shapiro, Stuart C.](#) (1995), "[Computationalism](#)" [postscript], *Minds and Machines* 5(4) (November): 517-524.
7. [Rapaport, William J.](#) (1998), "[How Minds Can Be Computational Systems](#)", *Journal of Experimental and Theoretical Artificial Intelligence* 10: 403-419.

Abstract: The proper treatment of computationalism, as the thesis that cognition is computable, is presented and defended. Some arguments of [James H. Fetzer](#) against computationalism are examined and found wanting, and his positive theory of minds as semiotic systems is shown to be consistent with computationalism. An objection is raised to an argument of [Selmer Bringsjord](#) against one strand of computationalism, viz., that Turing-Test-passing artifacts are persons; it is argued that, whether or not this objection holds, such artifacts will inevitably be persons.

8. [McDermott, Drew](#) (2001), *Mind and Mechanism* (Cambridge, MA: MIT Press).
 - Toyama, Kentaro (1996/2001), "[An Interview with Drew McDermott](#)", *ACM Crossroads* 3-1.
9. [Smith, Brian Cantwell](#) (2002), "The Foundations of Computing", in Scheutz, Matthias (ed.), *Computationalism: New Directions* (Cambridge, MA: MIT Press): 23-58.
 - For a critique of this article, see:
 - DeJohn, Jerry; & Dietrich, Eric (2003), "[Editorial: Subvert the Dominant Paradigm! A Review of Computationalism: New Directions, edited by Matthias Scheutz](#)", *Journal of Experimental & Theoretical Artificial Intelligence* 15(4) (October-December): 375-382, esp. §3.1, pp. 378-379.
 - [direct access](#) [not yet available; stay tuned]

Computer Ethics

Last Update: 13 April 2004

Note: **NEW** or **UPDATED** material is highlighted

Entire courses have been devoted to this topic. **For more information, do a Google search by clicking on the title above.** I also have a large file of articles and newspaper clippings; stop by my office if you want to browse through it.

Websites

1. AAAI's AI Topics website on Ethical and Social Implications of AI
 - A great site, with many links.
-

Readings

There are numerous books on computer ethics. For those at UB, type "computer ethics" as a Keyword into Bison.

1. * Lem, Stanislaw (1971), "Non Serviam", in S. Lem, *A Perfect Vacuum*, trans. by Michael Kandel (New York: Harcourt Brace Jovanovich, 1979).
 - Reprinted as:
"The Experiment (A Review of "Non Serviam", by James Dobb)" [PDF], *The New Yorker* (24 July 1978): 26ff.
 - Reprinted in:
Hofstadter, Douglas R.; & Dennett, Daniel C. (eds.) (1981), *The Mind's I: Fantasies and Reflections on Self and Soul* (New York: Basic Books): 296-317.
 - Also see:
Hofstadter, Douglas R.; & Dennett, Daniel C. (1981), "Reflections [on Lem's "Non Serviam"]", in Hofstadter, Douglas R.; & Dennett, Daniel C. (eds.) (1981), *The Mind's I: Fantasies and Reflections on Self and Soul* (New York: Basic Books): 317-320.
 - This is *must* reading for anyone interested in either Artificial Life or in the moral/ethical implications of AI.
 - Another "must read":

Powers, Richard (1995), *Galatea 2.2* (New York: Farrar, Straus, Giroux);
 LOCKWOOD Book Collection PS3566 .O92 G35 1995

2. * Moor, James H. (1979), "Are There Decisions Computers Should Never Make?" [PDF], *Nature and System* 1: 217-229.

Good articles to read in contrast to Moor's paper:

- Friedman, Batya; & Kahn, Peter H., Jr. (1992), "People Are Responsible, Computers Are Not" [PDF], excerpt from their "Human Agency and Responsible Computing: Implications for Computer System Design", *Journal of Systems Software* (1992): 7-14; excerpt reprinted in M. David Ermann, Mary B. Williams, & Michele S. Shauf (eds.) (1997), *Computers, Ethics, and Society, Second Edition* (New York: Oxford University Press): 303-314.

- The online version also includes pp. 313-314, containing "The Ten Commandments of Computer Ethics"
- Friedman & Kahn's main point is that programmers should not design computer systems so that users think that the systems are "intelligent". The April 2004 issue of *CACM* has a whole section devoted to this:

Miller, Christopher A. (guest ed.), "Human-Computer Etiquette: Managing Expectations with Intentional Agents", *Communications of the ACM* 47(4) (April): 30-61.

- Johnson, George (2002), "To Err Is Human", *New York Times* (14 July).
 - Provides an interesting real-life case study of Moor's problem.
- Brachman, Ronald J. (2002), "Systems that Know What They're Doing" [PDF], *IEEE Intelligent Systems* (November/December): 67-71.
 - Suggests how and why decision-making computers should be able to explain their decisions.
- Aref, Hassan (2004), "Recipe for an Affordable Supercomputer: Take 1,100 Apples...", *Chronicle of Higher Education* (5 March): B14.
 - Suggests (but does not discuss) that supercomputers might make decisions that we could not understand:

"As we construct machines that rival the mental capability of humans, will our analytical skills atrophy? Will we come to rely too much on the ability to do brute-force simulations in a very short time, rather than subject problems to careful analysis? Will we run to the computer before thinking a problem through?...A major challenge for the future of humanity is whether we can also learn to master machines that outperform us mentally."

- On the notion of "our analytical skills atrophy"ing, you might enjoy the following science-fiction story about a human who rediscovers how to do arithmetic after all arithmetical problems are handled by computers:

Asimov, Isaac (1957), "The Feeling of Power", reprinted in Clifton Fadiman (ed.), *The Mathematical Magpie* (New York: Simon and Schuster, 1962): 3-14.

- Kolata, Gina (2004), "New Studies Question Value of Opening Arteries" *The New York Times* (21 March): A1,A21.
 - A paragraph deeply embedded in this article suggests that people find it difficult to accept the rational recommendations even of other people. The article reports on evidence that a certain popular and common surgical procedure has just been shown to be of no benefit:

"Dr. Hillis said he tried to explain the evidence to patients, to little avail. "You end up reaching a level of frustration," he said. "I think they have talked to someone along the line who convinced them that this procedure will save their life." "

- 3. * Moor, James H. (1985), "What Is Computer Ethics?", *Metaphilosophy* 16(4) (October): 266-275.
 - HTML format

- 4. * LaChat, Michael R. (1986), "Artificial Intelligence and Ethics: An Exercise in the Moral Imagination" [PDF], *AI Magazine* 7(2): 70-79.
 - A good follow-up essay; argues that we should build robots that will be more moral than we are (and then we should "exit stage left"):

Dietrich, Eric (2001), "Homo sapiens 2.0: why we should build the better robots of our nature" [PDF], *Journal of Experimental and Theoretical Artificial Intelligence* 13(4) (October): 323-328.
 - A follow-up essay by LaChat, in which he argues that a "moral" robot "will have to possess sentient properties, chiefly pain perception and emotion, in order to develop an *empathetic* superego which human persons would find necessary and permissible in a morally autonomous AI":

La Chat, Michael Ray (2003), "Moral Stages in the Evolution of the Artificial Superego: A Cost-Benefits Trajectory", in Iva Smit, Wendell Wallach, & Goerge E. Lasker (eds.), *Cognitive, Emotive and Ethical Aspects of Decision Making in Humans and in Artificial Intelligence*, Vol. II (Windsor, ON, CANADA: International Institute for Advanced Studies in Systems Research and Cybernetics):18-24.

- 5. Turkle, Sherry (2004), "How Computers Change the Way We Think", *Chronicle of Higher Education* (January 30): B26-B28.

Philosophy of Computer Science

Course Summary

Last Update: 21 April 2004

Note: **NEW** or **UPDATED** material is highlighted

1. What is philosophy?

- Brief history of western philosophy as relevant to CS
 - My definition:
 - philosophy =def search for truth, in any field, by rational means
 - argument analysis ("argument", "premise", "conclusion", "valid", "factual", "sound")
 - Main branches of philosophy
 - Especially "philosophy of X" = study of main goals & fundamental assumptions of X
-

2. What is computer science?

- Motivations for asking the question (political, philosophical)
- Newell, Perlis, & Simon 1967:
 - CS = science of computers and surrounding phenomena
 - including algorithms, etc.
- Knuth 1974:
 - CS = study of algorithms
 - including the computers they run on, etc.
- Newell & Simon 1976:
 - CS = empirical study ("artificial science") of the phenomena surrounding computers
- Denning 1985:
 - CS = body of knowledge dealing with information-transforming processes
- Harmanis & Lin 1992:

- CS = study of information
 - Brooks 1996:
 - CS is engineering, not science.
 - Shapiro 2001:
 - CS = natural science of procedures
-

3. Is CS a science or engineering?

a. What is science?

- Goals = description vs. explanation
 - instrumentalism vs. realism
 - scientific method
 - Bacon (~1600): experimental method
 - Popper(~1950): conjectures & refutations
 - X is scientific iff X is falsifiable
 - empirical sciences vs. non-empirical sciences (e.g., math)
-

b. What is engineering?

- pure vs. applied sciences
 - Loui 1987:
 - CS = a new kind of engineering that studies:
 - theory, design, analysis, implementation of information-processing algorithms
 - Davis 1998: history of engineering
 - Engineering =? application of science for use & convenience of people and to improve means of production
 - Petroski 2003:
 - fundamental activity of engineering is **design**
-

4. What is a computer? -- I

- History of computers; 2 parallel goals:
 - goal 1 = to build a computing machine
 - Babbage, Aiken, Atanasoff & Berry, Turing, Eckert & Mauchly

- goal 2 = to provide a foundation for math
 - Leibniz, Boole, Frege, Hilbert, Turing, Church, Gödel
-

5. What is an algorithm? -- I

- what is computation?
 - A function f is computable =def there is an algorithm that computes f ; i.e., there is an algorithm A such that for all input i , $A(i)=f(i)$ and A specifies how f 's input and output are related.
 - algorithm for P =def a finite procedure (i.e., a finite set of instructions) for solving P that is:
 - a. unambiguous for the computer or human who will execute it; i.e., all steps of the procedure must be clear and well-defined for the executor, and
 - b. effective; i.e., it must eventually halt, and it must output a correct solution to P .
 - Turing Machines
 - history of term "computable"
 - Turing's Thesis: a function is computable iff it is TM-computable.
 - Church's Thesis: a function is computable iff it is lambda-definable (\equiv recursive)
-

6. What is a computer? -- II

- Searle, "Is the Brain a Digital Computer?"
 - Yes; because **everything** is a digital computer
 - Hayes et al. symposium
 - computer = machine that can take as I/P patterns that describe changes to themselves & other patterns, and that causes the described changes to occur
-

7. What is an algorithm? -- II

- Preston on difference between recipes and algorithms
 - recipes are more like specifications
 - Cleland:
 - Mundane procedures (causal processes, including recipes) are effective procedures that are not TM-computable
 - Effectiveness of mundane procedures depends on external world
-

8. What is hypercomputation?

- =def computation of functions that can't be TM-computed
 - Turing's "oracle" machines
 - Putnam's & Gold's "trial & error" machines (TM, but last answer counts)
 - Boolos & Jeffrey's "Zeus" machines (infinitely accelerating)
 - Wegner's "interaction" machines
 - Kugel 2002: Putnam-Gold machines may be needed for AI to succeed
-

9. What is a computer program?

a. What is implementation?

- Chalmers: implementation is isomorphism
 - argument against Searle
 - a computer is an implementation of a computation or of a UTM
 - Rapaport: implementation is semantic interpretation of an Abstraction in some medium
 - syntax vs. semantics of formal systems
-

b. Are programs scientific theories?

- programs are (a language for expressing) theories, which can then be their own models
 - theory vs. model, simulation vs. real thing, simulation vs. emulation
 - philosophical theories of scientific explanation
 - philosophical theories of scientific models
-

c. What is software?

- Moor: software is a computer program that is changeable by a person
 - Suber: software is syntactic form
 - Colburn: software is a concrete abstraction
 - has a "medium of description": text in a formal language (abstraction)
 - has a "medium of execution": circuits & semiconductors (concrete)
-

d. Can software be patented? Or should it be copyrighted?

- copyrights vs. patents
 - Newell: need to devise good models ("ontologies") of algorithms & other CS entities
 - Koepsell: need to revise the models of legal protection.
-

e. Can programs be verified?

- Smith, "Limits of Correctness"
 - There is a gap between the world and our models of it.
 - Computers rely on models of the models, but must act in the real world.
 - Fetzer: programs can't be verified
 - ...because you can't logically prove that causal systems won't fail.
 - at best, can verify an algorithm
-

10. Philosophy of AI: *Could we build artificial intelligences?*

- Turing Test:
 - A computer will be said to be able to think if we cannot distinguish a computer's cognitive behavior from a human's
 - Searle's Chinese Room Argument:
 - A computer could pass a Turing Test without really being able to think.
 - Rapaport's way out of the Chinese Room:
 - Syntax can suffice for semantic interpretation of the kind needed for computational cognition
-

11. Computer Ethics

- What is ethics?
 - Moor on "what is computer ethics":
 - Need to have metaphysical/ontological theories of computers and related phenomena in order to answer ethical and social questions about their nature and use.
 - In particular, computers are "logically malleable"
- Moor: "Are there decisions computers should never make?"

- No, at least, not as long as their track record is better than that of humans
- But it's up to us to accept/reject their decisions

- Friedman & Kahn: Yes, because only humans are capable of being moral agents

- But: "to err is human"--the case of the airline crash caused by following a human's decision instead of a computer's.

- **Should** we build artificial intelligences?
 - Lem: We may someday have to pull the plug
 - Lachat: Maybe we shouldn't even begin
 - But considering the possibilities enables us to deal with important issues:
 - what is a person?
 - would an AI with personhood have rights?
 - could it be moral?

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/course-summary-2004-04-21.html

CSE 4/510 & PHI 498, Spring 2004

Final Exam

Last Update: 22 April 2004

Note: **NEW** or **UPDATED** material is highlighted

This is a take-home, open-notes, open-books, open-articles, closed-peers exam.(*)

Survey the main answers, as presented and discussed in the readings and during the semester's lectures, to each of the following 5 questions, presenting, analyzing, and evaluating the arguments in favor of each answer.

You do not need to give short answers to each "sub"-question within each of the 4 main questions, and you probably should not simply try to do so. Rather, write a short essay that touches on all the topics in each main question.

Plan to write between 10 and 15 pages (preferably closer to 10; absolute maximum = 15 pp.)--i.e., about 2500-3750 words (figuring approximately 250 words per double-spaced, single-sided page). Hence, figure about 2-3 pp. (500-750 words) per question. (You have 1 week for the exam. I suggest spreading the writing over that week, spending perhaps one day per question, and a day or two for polishing it up.)

As with the term paper (for those of you choosing that option instead), the final document should be prepared in accordance with the instructions in my ["How to Write"](#) website. If you cite any material from any source other than your own notes, and especially if you use any quotations, please give a full citation, and include a bibliography (the bibliography will not count against your page limit).

Those of you who have chosen to do the term paper instead of the final exam may, at the "last" minute, switch to the final-exam option. However, you may not choose both options, and you must adhere strictly to the common deadline (see below).

1. What is computer science? (That is: Is it a science? If so, what is it a science of? Is it a science of computers? (What is a computer?) Or is it a science of computation?)
2. What is computation? (Computations are said to be algorithms, so what is an algorithm? Algorithms are said to be procedures, or recipes, so what is a procedure? What is a recipe? What are Church's and Turing's "theses"? What is "hypercomputation"?)
3. What is a computer program? (That is: What is the relation of a program to that which it models or simulates? What is simulation? Are programs (scientific) theories? Algorithms are said to be implemented in computer programs, so what is a computer program, and what is an implementation? What is software? Should software/algorithms/computer programs be copyrighted, or patented? Can computer programs be verified?)
4. What is the relation of computation to cognition? (That is: Can computers think? What are the

Turing Test and the Chinese Room Argument?)

5. Should we trust decisions made by computers? (For example: Are there decisions that computers should never make?) Should we build "intelligent" computers?

**BOTH THIS FINAL EXAM AND THE TERM PAPER
(AND ANY READING JOURNALS OR POSITION PAPERS THAT HAVE NOT YET
BEEN HANDED IN)
ARE DUE IN MY OFFICE (BELL 214) OR MY MAILBOX (BELL 211)
BY 5:00 P.M., THURSDAY, MAY 6.
NO LATE EXAMS, TERM PAPERS, READING JOURNALS, OR POSITION PAPERS
WILL BE ACCEPTED.**

() I.e., don't "peer" at your peers' exams!*

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/2004-04-20.html

Term-Paper Topics

Last Update: 15 February 2004

Note: **NEW** or **UPDATED** material is highlighted

Here is a list of some possible term-paper topics:

1. Further discussion of any topic covered in class (e.g., your answer to one of the questions listed on the syllabus, plus a defense of your answer, *or* a critical examination of someone else's (published) answer to one of the questions).
2. A critical examination of any of the required or recommended (or any other approved and relevant) readings.
3. A critical study of any monograph (i.e., book) or anthology (including special issues of journals) on the philosophy of computer science.
4. A critical, but general, survey article on the philosophy of computer science that would be appropriate for an encyclopedia of philosophy or an encyclopedia of computer science.
 - Variation: A presentation and well-argued defense of *your* "philosophy of computer science", i.e., *your* answers to all (or most) of our questions, together with supporting reasons.
5. Other ideas of your own, approved by me in advance.

For general assistance with writing (including my required method of paper preparation and format, as well as advice on grammar), see my website "[How to Write](#)".

For specific assistance on writing a philosophy paper, see the text by Woodhouse or any of the guides on the Google list accessible from the "[What Is Philosophy?](#)" Webpage.

The paper should be a maximum of 10-15 **NEW** **double-spaced, single-sided** pages (i.e., about 2500-3750 words) (not counting the bibliography).

Reminder: An abstract and reading list are due no later than Tuesday, February 24.

Position-Paper Assignments

Last Update: 20 April 2004

Note: **NEW** or **UPDATED** material is highlighted

1. Position Paper #5: Can Computers Think?

- Assigned: 13 Apr 2004
- 5 copies due: 20 Apr 2004
- **NEW** Suggestions and Guidelines for Peer-Group Editing of Position Paper #5

2. Position Paper #4: Computer Ethics

- Assigned: 1 Apr 2004
- 5 copies due: 8 Apr 2004
- Suggestions and Guidelines for Peer-Group Editing of Position Paper #4

3. Position Paper #3: What Is a Computer Program?

- Assigned: 11 Mar 2004
- 5 copies due: 25 Mar 2004
- Suggestions and Guidelines for Peer-Group Editing of Position Paper #3

4. Position Paper #2: What Is Computation?

- Assigned: 17 Feb 2004
- 5 copies due: 24 Feb 2004
- Suggestions and Guidelines for Peer-Group Editing of Position Paper #2

5. Position Paper #1: What Is Computer Science?

- Assigned: 22 Jan 2004
- 5 copies due: 29 Jan 2004
- Suggestions and Guidelines for Peer-Group Editing of Position Paper #1

Position Paper #1: What Is Computer Science?

Last Update: 22 January 2003

Note: **NEW** or **UPDATED** material is highlighted

The purpose of this position paper is to give you an opportunity to clarify *your* beliefs about what computer science is, so that as we continue to discuss the topic in class and as you continue to read about it, you'll know where *you* stand--what *your* beliefs are. Later, when your beliefs have been "contaminated" by further readings and by our discussions, you may wish to *revise* your beliefs. But you can't revise a belief that you don't have (you can only acquire new beliefs). So, here I am forcing you to *discover*, *clarify*, and *defend* the beliefs that you *now* have, by turning them into words and putting them on paper.

Imagine that you are the newly-appointed Dean of the School of Science at the University of Aix (pronounced like the letter "X"). In an attempt to build up the rival School of Engineering, the newly-appointed Dean of Engineering has proposed to the Provost (the Deans' boss) that the Department of Computer Science be moved--lock, stock, and computer, so to speak--to Engineering, on the following grounds:

1. Science is the systematic observation, description, experimental investigation, and theoretical explanation of natural phenomena.
2. Computer science is the study of computers and related phenomena.
3. Therefore, computer science is not a science.

(The Dean of Engineering has not *yet* argued that computer science is an engineering discipline; that may come later.)

You may agree with this argument; then again, you may not agree with it. You should ignore political considerations: you may suppose that the move from Science to Engineering involves no loss of money, prestige, or anything else, and it is to be done, if at all, only on strictly intellectual grounds. How might you respond to the Dean of Engineering's argument? The Provost is eagerly awaiting your reply, and will abide by your decision...*if*, that is, you give a well-argued defense of your position.

There are several possible responses that you might have:

Response 1:

You might **disagree** for any of 3 reasons:

- a. You believe that premise (1) is false.
- b. You believe that premise (2) is false.

- c. You believe that (1) and (2) are true but that conclusion (3) does not follow from them.
- E.g., you might believe that "computers and related phenomena" *are* "natural phenomena".

...or you might believe some combination of these.

Please explain to the Provost *why* you disagree, by explaining which of (a), (b), and/or (c) you *do* believe.

Response 2:

You might **agree** for any of the following reasons:

- a. You believe (1) and (2) and that (3) follows from them. If so, please explain to the Provost why you believe (1) and (2), and how (3) follows from them. (E.g., you might believe that "computers and related phenomena" are *not* "natural phenomena", or you might believe that computer science doesn't study them "systematically", or....)
- b. You don't believe (1) or you don't believe (2), but you *do* believe (3), whether or not it follows from (1) and (2). If so, please explain why you don't believe (1) and/or (2) and what *other* reasons you have for believing (3).

Response 3:

You might **neither** agree **nor** disagree with (3); alternatively, you might **both** agree **and** disagree with it. For example, you might believe that computer science is *both* a science *and* an engineering discipline (or, alternatively, that it is neither). If so, then please give your reasons for this.

Other responses:

You might not agree with any of these responses. However, I believe that any other response can, perhaps with a bit of force, be seen to fall under one of the above Responses. But if you really feel that your position is not exactly characterized by any of the above Responses, then please say what your position is, why you believe it, and why you think it is not one of the above.

Ground Rules:

1. If you resort to a dictionary, textbook, article, website, etc., be sure to say which one.
2. Your answer should honestly reflect *your* beliefs (not what you think the fictional Provost or Dean of Engineering want to hear!).
3. Your position paper should be approximately

1 typed page and double-spaced (i.e., about 250 words).
4. Please bring

5 copies

 to class on the due date.
5. This paper only needs the title "Position Paper #1", your name, and the date at the top of the page.

6. For general assistance with writing (including my preferred method of paper preparation and format, as well as advice on grammar), see my website "[How to Write](#)".

DUE AT THE BEGINNING OF LECTURE, THURSDAY, JANUARY 29

Copyright © 2004 by [William J. Rapaport](#) (rapaport@cse.buffalo.edu)
file: 510/pospaper1.2004.01.20.html

Suggestions and Guidelines for Peer-Group Editing of Position Paper #1

Last Update: 23 January 2003

Note: **NEW** or **UPDATED** material is highlighted

1. When you get into your small groups, introduce yourselves, and share copies of your papers with each other.
2. Choose one paper to discuss first. (Suggestion: Go in alphabetical order by family name.)
3. The other people in the group might find it useful to imagine themselves as members of a committee set up by the Provost to make a recommendation. Their purpose is to try to help the author clarify his or her beliefs and arguments, so that they will be able to make a recommendation to the Provost on purely logical grounds (again: ignore politics!).
4. Start by asking the author to state (or read) his or her beliefs about whether computer science is a science, giving his or her *reasons* for those beliefs.
5. Any time you have a question, ask it. Here are some suggestions:
 - Why did you say ___ rather than ___?
 - What did you mean when you said ___?
 - Can you give me an example of ___?
 - Can you give me more details about ___?
 - Do you think that ___ is always true?
 - Why? (This is always a good question to ask.)
 - How?
6. The author should not get defensive. The committee members are friendly. Critical, but friendly.
7. Keep a written record of the questions and replies. This will be useful to the author, for revision.
8. After spending *at least* 10 minutes on the first paper, move on to the next, going back to step (2) above, changing roles. Spend about 15 minutes per paper.
9. At home, over the next week, please *revise* your paper to take into consideration the comments made by your fellow students (i.e., your "peers"): Perhaps defend your claims better, or clarify statements that were misunderstood, etc. For help, see Dima or me.

**1-2 PAGE (250-500 WORD) REVISION, 1 COPY, TYPED, IS DUE AT THE BEGINNING
OF LECTURE, THURSDAY, FEB. 5.
NO LATE PAPERS WILL BE ACCEPTED!**

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/peered1.2004.01.23.html

Position Paper #2: What Is Computation?

Last Update: 17 February 2004

Note: **NEW** or **UPDATED** material is highlighted

For this position paper, I would like you to evaluate the following argument:

1. An algorithm, A , for solving a problem P (or, for computing a function F) is a **finite procedure** (i.e., a finite set of instructions) for solving P (or, computing F) that is:
 - a. **unambiguous** for the computer or human who will execute it;
 - i.e., all steps of the procedure must be clear and well-defined for the executor
 - and
 - b. **effective**; i.e., A must eventually halt, and it must output a correct solution to P (or, the correct computation of F).
2. Computer programming languages (like Java, Lisp, Fortran, etc.) are formal languages for computing algorithms.
3. Every computer programming language is equivalent in expressibility to a Turing-machine programming language.
 - a. I.e., every program in any programming language can be translated into the language for programming Turing machines, and vice versa.
 - b. I.e., any problem that is solvable by any programming language (or, any function that is computable by any programming language) is solvable (or, computable) by a Turing machine, and vice versa.
4. Some real computer programs violate parts of definition 1:
 - a. E.g., airline-reservation systems, ATMs, operating systems, etc., never halt.
 - b. E.g., heuristic AI programs don't always solve exactly the problem that they were written for, but only come very close.
 - c. E.g., the effectiveness of mundane procedures depends on the environment in which they are executed. (And so on.)
5. Therefore, these programs cannot be modeled by Turing machines (contrary to premise 3).
6. Therefore, they are not computable. (But how can a real computer program not be computable?!)

To evaluate this argument, you must determine whether it is "**factual**" and whether it is **valid**.

- If it is both, then it is said to be **sound**.
- You are logically obligated to believe the conclusions of sound arguments! So, if you ever come across an argument that you think is sound, but whose conclusion you don't believe, then either it is "unfactual" [i.e., one or more of the premises are false] or it is invalid [i.e., there is some way for the premises to be true yet for the conclusion to be false].

To determine whether it is "factual", you must decide whether each premise is true or false (more realistically, you must decide whether you believe, or agree with, each premise), **and** you must explain why or why not.

To determine whether it is valid, you must suppose "for the sake of the argument" that all the premises *are* true, and then consider whether the conclusions logically follow from them. (Or: Can you imagine some way the world might be so that the premises are true but the conclusion is false?)

Note that there are *two* conclusions: lines 5 and 6. So, do you agree that conclusion 5 follows logically from premises 1-4 and/or that conclusion 6 follows logically from 5? If not, are there missing premises that are needed to make the argument(s) valid? If there are, do you agree with them (why/why not)?

Finally, do you agree with the conclusion(s)? If you do, but you think that there's something wrong with the argument, try to present a better one. If you don't agree with the conclusion(s), state why, and try to give an argument for what you *do* believe.

-
- Your position paper should be approximately **1-2 typed pages, double-spaced (i.e., about 250-500 words), and single-sided.**
 - Please bring **5 copies** to class on the due date.
 - This paper only needs the title "Position Paper #2", your name, the date, and the course number (410, 498, or 510) at the top of the page.
 - For general assistance with writing (including my preferred method of paper preparation and format, as well as advice on grammar), see my website "[How to Write](#)".

DUE AT THE BEGINNING OF LECTURE, TUESDAY, FEBRUARY 24

Suggestions and Guidelines for Peer-Group Editing of Position Paper #2

Last Update: 22 February 2004

Note: **NEW** or **UPDATED** material is highlighted

1. Distribute your papers among the group members.
2. Spend 10-15 minutes on each paper. (If we start at about 9:45, you'll have about 65 minutes. If there are 2 papers in your group, you can spend about 30 minutes on each; if there are 3, spend about 20 minutes on each; if there are 4, spend about 15 minutes on each.)
 - a. It's better if there are at least 3 papers per group, so, if there aren't, please let me know.
3. For each paper, ask as many of the following questions as you have time for:
 - a. Did the author state whether **and** why they did or did not agree with the definition in premise 1?

For premise 1 and each of the rest of the premises, also ask these questions:

 - i. If the author **agreed**, then it is preferable (but not necessary) that they give reasons for agreeing. If they did give such reasons, do **you** agree with those reasons? Why?
 - ii. If the author **disagreed**, it **is** necessary that they give reasons for disagreeing, so do **you** agree with those reasons? Why?
 - b. Did the author state whether and why they did (not) agree with the claim about the nature of programming languages in premise 2?

(Plus questions (i) and (ii), above.)
 - c. Did the author state whether and why they did (not) agree with the claim about the "Turing-equivalence" of programming languages in premise 3?

(Plus questions (i) and (ii), above.)
 - d. Did the author state whether and why they did (not) agree about the claim and/or the examples in premise 4?

(Plus questions (i) and (ii), above.)
 - e. Did the author state whether and why they believed that conclusion 5 did (not) follow from premises 1-4? Do you agree?
 - i. If the author believed that this conclusion did **not** logically follow from the premises,

did they state whether they believed it anyway, on independent grounds (i.e., for different reasons)?

ii. And, if so, do you agree with those reasons?

f. Did the author state whether and why they believed that conclusion 6 did (not) follow from statement 5? Do you agree?

(Plus questions (e)(i) & (ii), above.)

4. Keep a written record of the questions and replies. This will be useful to the author, for revision.

5. At home, over the next week, please *revise* your paper to take into consideration the comments made by your fellow students (i.e., your "peers"): Perhaps defend your claims better, or clarify statements that were misunderstood, etc. For help, see Dima or me.

**1-2 PAGE (250-500 WORD), TYPED, DOUBLE-SPACED, SINGLE-SIDED REVISION, 1 COPY, IS DUE AT THE BEGINNING OF LECTURE, TUESDAY, MAR. 2.
NO LATE PAPERS WILL BE ACCEPTED!**

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/peered2.2004.02.22.html

Position Paper #3: What Is a Computer Program?

Last Update: 11 March 2004

Note: **NEW** or **UPDATED** material is highlighted

For this position paper, I would like you to evaluate the following argument:

1. A special-purpose computer (i.e., a computer that does just one task) is essentially a hardwired computer program.
2. Such a hardwired computer program is a machine.
3. Machines can be patented.
4. Therefore, such a hardwired computer program can be patented.
5. The printed text of a computer program is a "literary work" (i.e., a piece of writing).
6. Literary works can be copyrighted.
7. Therefore, such a computer program can be copyrighted.
8. Nothing can be both patented and copyrighted.
 - **Note:** This premise is a matter of law. You must accept it as true.
9. There is no computational or other relevant difference between the hardwired computer program and its textual counterpart (except for the different media in which they are implemented, one being hardwired and the other being written on, say, a piece of paper).
10. Therefore, computer programs can be both patented and copyrighted.

To help you evaluate this argument (which we'll look at in more detail in lecture later this semester), here are some extracts from some relevant websites:

- a. From the official US Patent Office definition of "patent":

a property right granted by the Government of the United States of America to an inventor "to exclude others from making, using, offering for sale, or selling the invention throughout the United States or importing the invention into the United States" for a limited time in exchange for public disclosure of the invention when the patent is granted.

- b. The Patent Office definition of "invention":

any art or process (way of doing or making things), machine, manufacture, design, or composition of matter, or any new and useful improvement thereof, or any variety of plant, which is or may be patentable under the patent laws of the United States.

c. The official US Copyright Office definition of "copyright":

Copyright is a form of protection provided by the laws of the United States...to the authors of "original works of authorship," including literary, dramatic, musical, artistic, and certain other intellectual works. This protection is available to both published and unpublished works.

d. From the same website:

Copyrightable works include the following categories:

1. literary works;
2. musical works, including any accompanying words
3. dramatic works, including any accompanying music
4. pantomimes and choreographic works
5. pictorial, graphic, and sculptural works
6. motion pictures and other audiovisual works
7. sound recordings
8. architectural works

These categories should be viewed broadly. For example, computer programs and most "compilations" may be registered as "literary works"; maps and architectural plans may be registered as "pictorial, graphic, and sculptural works."

WHAT IS NOT PROTECTED BY COPYRIGHT?

Several categories of material are generally not eligible for federal copyright protection. These include among others:

- Works that have not been fixed in a tangible form of expression (for example, choreographic works that have not been notated or recorded, or improvisational speeches or performances that have not been written or recorded)
- Titles, names, short phrases, and slogans; familiar symbols or designs; mere variations of typographic ornamentation, lettering, or coloring; mere listings of ingredients or contents
- Ideas, procedures, methods, systems, processes, concepts, principles, discoveries, or devices, as distinguished from a description, explanation, or illustration
- Works consisting entirely of information that is common property and containing no original authorship (for example: standard calendars, height and weight charts, tape measures and rulers, and lists or tables taken from public documents or other common sources)

-
- Your position paper should be approximately **1-2 typed pages, double-spaced (i.e., about 250-500 words), and single-sided.**
 - Please bring **5 copies** to class on the due date.
 - This paper only needs the title "Position Paper #3", your name, the date, and the course number

(410, 498, or 510) at the top of the page.

- For general assistance with writing (including my preferred method of paper preparation and format, as well as advice on grammar), see my website "[How to Write](#)".
- For general guidelines on how to evaluate an argument, see the newsgroup posting in "[Subject: POSITION PAPER 2 -- ANALYSIS & GRADING](#)" or the website "[Critical Thinking Core Concepts](#)" (also see its [table of contents](#)).

DUE AT THE BEGINNING OF LECTURE, THURSDAY, MARCH 25

Copyright © 2004 by [William J. Rapaport](#) (rapaport@cse.buffalo.edu)
file: 510/pospaper3-2004-03-09.html

Suggestions and Guidelines for Peer-Group Editing of Position Paper #3

Last Update: 25 March 2004, 8:30 p.m.`

Note: **NEW** or **UPDATED** material is highlighted

1. Distribute your papers among the group members.
2. Spend 10-15 minutes on each paper. (If we start at about 9:45, you'll have about 65 minutes. If there are 2 papers in your group, you can spend about 30 minutes on each; if there are 3, spend about 20 minutes on each; if there are 4, spend about 15 minutes on each.)
 - It's better if there are at least 3 papers per group, so, if there aren't, please let me know.
3. Here is my (incomplete) analysis:
 - a. The overall argument consists of 3 "sub"arguments:
 - i. 1,2,3; therefore, 4
 - ii. 5,6; therefore, 7
 - iii. 4,7,9; therefore, 10
 - b. All of them are valid (i.e., it is impossible for the premises to be true and the conclusion to be false).
 - c. But 10 conflicts with 8, which is true.
 - d. Therefore, 10 is false.
 - e. Therefore, at least one of 4, 7, 9 is false!
 - f. But if 4 is false, then at least one of 1, 2, 3 is false!
 - g. And if 7 is false, then at least one of 5, 6 is false!
 - h. So, which ones are false? And why do you think so?
 - i. **NEW** Alternatively, if you are firmly convinced, for good reason, that 1,2,3,5,6 are all true, then you must think that the law (as expressed in 3,6, and especially 8) must be changed. How should it be changed?
 - j. **NEW** Note that (at least on my reading of them) Newell 1985-1986 argues that at least one of 1,2,3,5,6 is false (i.e., "the models are broken"), while Koepsell 2000 argues that the law needs to be changed.
4. For each paper, ask as many of the following questions as you have time for:
 - a. Did the author correctly identify the 3 subarguments? If not, did the author have a good reason for analyzing it differently?
 - b. Did the author (correctly) evaluate the validity of the (sub)argument(s)?
 - c. Did the author evaluate the truth values of the premises and give reasons for his or her

evaluations?

5. Keep a written record of the questions and replies. This will be useful to the author, for revision.
6. At home, over the next week, please *revise* your paper to take into consideration the comments made by your fellow students (i.e., your "peers"): Perhaps defend your claims better, or clarify statements that were misunderstood, etc. For help, see Dima or me.

1-2 PAGE (250-500 WORD), TYPED, DOUBLE-SPACED, SINGLE-SIDED REVISION, 1 COPY, IS DUE AT THE BEGINNING OF LECTURE, THURSDAY, APR. 1. NO LATE PAPERS WILL BE ACCEPTED!

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/peered3-2004-03-25-2.html

Position Paper #4: Are There Decisions Computers Should (Not) Make?

Last Update: 25 March 2004

Note: **NEW** or **UPDATED** material is highlighted

For this position paper, I would like you to evaluate the following argument:

1. Certain computers (i.e., computers running certain computer programs) can make rational decisions.
 - a. That is, they can determine the validity of arguments and ascertain the probable truth-values of the premises of the arguments, and they can consider the relative advantages and disadvantages of different courses of action to determine the best possible choices.
 - b. For example, there are computers that can diagnose certain diseases and (presumably) recommend appropriate medical treatments.
2. Suppose for the sake of argument that some of these computers can make decisions (and recommendations) on certain important matters concerning human welfare.
3. Suppose further that they can regularly make *better* recommendations than human experts on these matters.
4. Therefore, these computers should make decisions on these important matters concerning human welfare.

- Your position paper should be approximately **1-2 typed pages, double-spaced (i.e., about 250-500 words), and single-sided.**
- Please bring **5 copies** to class on the due date.
- This paper only needs the title "Position Paper #4", your name, the date, and the course number (410, 498, or 510) at the top of the page.
- For general assistance with writing (including my preferred method of paper preparation and format, as well as advice on grammar), see my website "[How to Write](#)".
- For general guidelines on how to evaluate an argument, see the newsgroup posting in "[Subject: POSITION PAPER 2 -- ANALYSIS & GRADING](#)" or the website "[Critical Thinking Core Concepts](#)" (also see its [table of contents](#)).

DUE AT THE BEGINNING OF LECTURE, THURSDAY, APRIL 8

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/pospaper4-2004-03-25.html

Suggestions and Guidelines for Peer-Group Editing of Position Paper #4

Last Update: 6 April 2004

Note: **NEW** or **UPDATED** material is highlighted

1. Distribute your papers among the group members.
2. Spend 10-15 minutes on each paper. (If we start at about 9:45, you'll have about 65 minutes. If there are 2 papers in your group, you can spend about 30 minutes on each; if there are 3, spend about 20 minutes on each; if there are 4, spend about 15 minutes on each.)
 - It's better if there are at least 3 papers per group, so, if there aren't, please let me know.
3. **Hint:** This argument has a missing premise that will make it valid. But you will still have to decide whether you agree with that missing premise, and say why (i.e., you will have to give your own argument in favor of, or opposed to, the missing premise). For all practical purposes, you can assume that the other premises are true. They may, as a matter of fact, be false; but that falsity doesn't affect the validity of the argument, which is really about what *might* be the case in the future *if* all the explicit premises were true.

One point that some of you may still not be entirely clear on: What if you disagree with the missing premise? Does that mean that the conclusion of this argument is false (in your opinion)? **Not necessarily!**

If the argument (with the missing premise added) is valid, and if you agree with the missing premise (and all the others, at least for the sake of the argument), then you *must*--logically must--believe the conclusion.

But if the argument (with the missing premise added) is valid, and if you *don't* agree with the missing premise, it does *not* logically follow that the conclusion must be false! All that follows is that this is a bad argument (in your opinion) for that conclusion. A *bad* argument *for* a conclusion is not the same as a *good* argument *against* that conclusion!

So, if you didn't agree with the missing premise, then you have one more task: You need to decide if you agree or disagree with the conclusion *for some other reason*, and you must state that reason, preferably in the form of an argument with premises.

4. For each paper, ask as many of the following questions as you have time for:
 - a. Did the author identify a missing premise?
 - b. Does the missing premise make the argument valid?
 - c. Does the author state whether s/he agrees with the missing premise?

- d. Does the author give reasons for that belief?
 - e. Does the author state whether s/he agrees with the conclusion?
 - f. If the author disagreed with the missing premise, did s/he give better reasons for believing or disbelieving the conclusion?
5. Keep a written record of the questions and replies. This will be useful to the author, for revision.
 6. At home, over the next week, please *revise* your paper to take into consideration the comments made by your fellow students (i.e., your "peers"): Perhaps defend your claims better, or clarify statements that were misunderstood, etc. For help, see Dima or me.

**1-2 PAGE (250-500 WORD), TYPED, DOUBLE-SPACED, SINGLE-SIDED REVISION, 1 COPY, IS DUE AT THE BEGINNING OF LECTURE, THURSDAY, APR. 15.
NO LATE PAPERS WILL BE ACCEPTED!**

Copyright © 2004 by [William J. Rapaport](mailto:rapaport@cse.buffalo.edu) (rapaport@cse.buffalo.edu)
file: 510/peered4-2004-04-06.html

Position Paper #5: Can Computers Think?

Last Update: 25 March 2004

Note: **NEW** or **UPDATED** material is highlighted

For this position paper, I would like you to evaluate the following hypothetical debate.

Pro: If something behaves in all relevant ways as if it were cognitive, then it *is* cognitive.

Con: What do you mean by "being cognitive"?

Pro: I mean that it can perceive (see, hear, etc.); has beliefs, desires, and intentions; can remember; can use and understand natural language; can reason and make rational decisions; etc. You know, the sort of thing that AI researchers are trying to achieve by computational means.

Con: Do you think they will succeed?

Pro: I'm optimistic: I think that a suitable AI program (or maybe a suite of programs) will eventually behave in all these ways.

Con: But that means that you think that such an AI program *will be* cognitive?

Pro: Yes.

Con: But that's crazy! Computer programs are purely syntactic!

Pro: Now it's my turn to ask for clarification: What do you mean by "syntactic"?

Con: I mean that all it can do is to manipulate the symbols of a formal symbol system.

Pro: So what's the problem?

Con: The problem is that cognition is semantic! That is, it involves the semantic interpretation of those symbols.

Pro: Well, I'm not so sure about that. But suppose you're right. What then?

Con: Well, syntax does not suffice for semantics. So, no purely syntactic computer program can exhibit semantic cognition, even if it behaves in all relevant ways as if it were cognitive.

-
- It may help if you rewrite Pro's and Con's arguments in terms of premises and conclusions, and then evaluate those arguments.
 - Your position paper should be approximately

1-2 typed pages, double-spaced (i.e., about 250-500 words), and single-sided.

- Please bring **5 copies** to class on the due date.
- This paper only needs the title "Position Paper #5", your name, the date, and the course number (410, 498, or 510) at the top of the page.
- For general assistance with writing (including my preferred method of paper preparation and format, as well as advice on grammar), see my website "[How to Write](#)".
- For general guidelines on how to evaluate an argument, see the newsgroup posting in "[Subject: POSITION PAPER 2 -- ANALYSIS & GRADING](#)" or the website "[Critical Thinking Core Concepts](#)" (also see its [table of contents](#)).

DUE AT THE BEGINNING OF LECTURE, TUESDAY, APRIL 20

Copyright © 2004 by [William J. Rapaport](#) (rapaport@cse.buffalo.edu)
file: 510/pospaper5-2004-03-25.html

Suggestions and Guidelines for Peer-Group Editing of Position Paper #5

Last Update: 20 April 2004

Note: **NEW** or **UPDATED** material is highlighted

1. Distribute your papers among the group members.
2. Spend 10-15 minutes on each paper. (If we start at about 9:45, you'll have about 65 minutes. If there are 2 papers in your group, you can spend about 30 minutes on each; if there are 3, spend about 20 minutes on each; if there are 4, spend about 15 minutes on each.)
 - It's better if there are at least 3 papers per group, so, if there aren't, please let me know.
3. **Suggestion:** There are really 2 arguments in this dialogue: Pro's and Con's. So, the first task is to present each argument. Once you have identified the premises (including hidden premises) and conclusion of each argument, you can then analyze it for factuality and validity.
4. For each paper in your peer-editing group, ask as many of the following questions as you have time for:
 - a. Did the author present both Pro's and Con's arguments?
 - b. For each argument, did the author state whether and why s/he believes the argument to be valid?
 - Note: It's possible to formulate both arguments so that they *are* valid! (In which case, ascertaining their factuality becomes your central task.)
 - c. For each argument, did the author state whether and why s/he agrees with the premises?
 - d. For each argument, if the author believed either that the argument was invalid (even with missing premises added--i.e., that there was *no* way to make the argument valid) or that one or more of the premises was false, then did the author state whether and why s/he agrees with the conclusion?
5. Keep a written record of the questions and replies. This will be useful to the author, for revision.
6. At home, over the next week, please *revise* your paper to take into consideration the comments made by your fellow students (i.e., your "peers"): Perhaps defend your claims better, or clarify statements that were misunderstood, etc. For help, see Dima or me.

1-2 PAGE (250-500 WORD), TYPED, DOUBLE-SPACED, SINGLE-SIDED REVISION, 1 COPY, IS DUE ON TUESDAY, APR. 27, IN MY OFFICE (BELL 214) OR MY MAILBOX (BELL 211), BY 5:00

P.M.
NO LATE PAPERS WILL BE ACCEPTED!

*Copyright © 2004 by [William J. Rapaport](#) (rapaport@cse.buffalo.edu)
file: 510/peered5-2004-04-20.html*