UNDEBUGGABILITY AND COGNITIVE SCIENCE

A resource-realistic perspective suggests some indispensable features for a computer program that approximates all human mentality. The mind's program would differ fundamentally more from familiar types of software. These features seem to exclude reasonably establishing that a program correctly and completely models the mind.

CHRISTOPHER CHERNIAK

A central tenet of much recent work in cognitive science is a commitment to realism about the agent's resources. Perhaps there is something mean-spirited about challenging presuppositions that human beings have unbounded potentialities, but a theory, however normative or regulative, that proceeds on an assumption that people have God's brain seems at least interestingly incomplete. What is sought are models that are both computationally and psychologically realistic, as well as neurally and genetically realistic.

Starting with such a resource-realistic framework, this article explores some properties we can now expect to be inextricable from any computer superprogram that purports to represent all of human mentality. A complete computational approximation of the mind would be a (1) huge, (2) "branchy" and holistically structured, and (3) quick-and-dirty (i.e., computationally tractable, but formally incorrect/incomplete) (4) kludge (i.e., a radically inelegant set of procedures). The mind's program thus turns out to be fundamentally dissimilar to more familiar software, and software, in general, to be dissimilar to more familiar types of machines. In particular, these properties seem inherently to exclude our reasonably establishing that we have the right program. In this way, the full mind's program appears to be a type of practically unknowable thing-in-itself.

Generally, cognitive science seems to presuppose the manageability and feasibility of such a total mind's program. For example, one of the current standard textbooks of artificial intelligence (AI) begins, "The ultimate goal of AI research (which we are very far from achieving) is to build a person, or, more humbly, an animal" [12, p. 7]; that is, to construct a program for the *entire* creature. A standard ground plan is then enumerated, including input modules for vision and language, then deduction, planning, explanation, and learning units, and, finally, output modules for robotics and speech. An important example from the philosophical side of cognitive science has been Fodor's thesis that psychological processes are typically computational, and so that much human behavior can only be explained by a computational model of cognition (see, e.g., [21], especially chap. 1).

A strong tendency to posit "impossibility engines"—profoundly unfeasible mental mechanisms—has not been confined to the most abstract reaches of philosophy. It seems to be a pervasive, unnoticed, but problematic element of methodology throughout mind/brain science, even when that field is explicitly taken to extend all the way down to neurophysiology and neuroanatomy. The most extreme philosophical case I know of is the deductive ability presupposed by conventional rationality idealizations.[1] Standard rationality models require the agent to be some sort of perfect logician; in

Earlier drafts of this article were read as a public lecture at Wichita State University, Wichita, Kansas, October 1986; at the University of Minnesota Center for Philosophy of Science, Minneapolis, February 1987; and at the AAAI/UMIACS Workshop on Theoretical Issues in Conceptual Information Processing, Washington, D.C., June 1987.

[1] I have discussed this example in [13, chaps. 1 and 4].

particular, to be able to determine in finite time whether or not any given formal sentence is a first-order logical consequence of a given set of premises. Half a century ago, however, Church's theorem showed that the predicate calculus was undecidable—that is, no *algorithm* for this task is possible. What surprised me most about this simple observation was at a metalevel: I could find no previous discussion of the point. Now, what would explain not even raising the question, "Does the ideal agent's perfect logical ability have to exceed even that covered by the classical unsolvability theorems?" One explanation would be the ultimate inattention to scale: not distinguishing between an agent under the usual constraints of the absolute unsolvability results—finite space and (for each input) run time—and some more highly idealized reasoner that could not even in principle employ any algorithm.

A less extreme philosophical example of inattention to scale of resources can be seen in some of the computational costs implicit in perfect "charity" of interpretation along conventional Quine–Davidson lines,[2] where the fundamental methodological principle is that (except for "corrigible confusions") an agent must be regarded as maintaining perfect deductive consistency. Figure 1 reviews the literally cosmic resources consumed by the agent merely maintaining full truth-functional consistency, a "trivially" decidable problem. Quite moderate cases of the task would computationally hog-tie a rather physically ideal truth-table machine to an extent so vast that we have no familiar names for the numbers involved.

Our basic methodological instinct, at whatever explanatory level, seems to be to work out a model of the mind for "typical" cases—most importantly, very small instances—and then implicitly to suppose a grand induction to the full-scale case of a complete human mind. Cocktail-party anthropology would deride Hottentots for allegedly counting "1, 2, many"; the methodology of mind/brain science seems unthinkingly to approach "1, 2, 3, infinity." The unrealism regarding resources that I am suggesting is still pervasive is an inattention to scale, in particular, to consequences of scaling up models of the mind/brain. (As a more moderate example, this scale-up difficulty lurks in the familiar microworlds approach to the problem of knowledge representation in AI—that is, the divide-and-conquer strategy of beginning with small, tightly constrained domains of total human common sense for formalization.[3]) Perhaps overlooking limits to growth of models is part of our Cartesian heritage, stemming from the picture of mind as a substance that has spatial dimensions no more than do mathematical structures.

[2] For explanation of the idea of charity of interpretation of an agent, see [18] and [47].

[3] Dreyfus has attacked the microworlds methodology [20] along lines different from the present discussion. Among other things, he argues that total human commonsense knowledge cannot typically be separated into such autonomous zones and that commonsense understanding is indefinitely articulable. It is, so to speak, "everywhere dense," like the real-number plane: Between any two areas, or points, there is always another area.

BIG TEXTS

Suppose we take seriously the working hypothesis of computational psychology, that a major part of the human mind is a cognitive system of rules and representations, captured by some programlike object (one paradigmatic model would be Fodor's language of thought hypothesis). One way of looking at this framework assumption is simply to note that this program, as a syntactic object, is text that has a size. That is, we can ask, "how big is the mind's program?"[4] It may seem unnatural to ask such a question if one does not distinguish between a concrete algorithm and the corresponding abstract function (asking the size of a mathematical function seems as much a category mistake as asking how much the number 15 weighs). Moreover, individuating such cognitive elements as beliefs and concepts is a notoriously messy business.

This "mind as warehouse" approach is rather primitive, but some evidence that the mind's storage capacities are not unlimited can be found in the renowned overflow phenomena of information explosion. For in-

Test of truth-functional consistency of the belief set
by the truth-table method, and
by an ideal computer I:

I checks each line of the truth table in one "supercycle":
the time a light ray takes to traverse the diameter of a proton.

Speed of light = 299,726 km/s, and the proton is 10^{-13} cm in diameter;
so, supercycle = 2.9×10^{-23} s.

Maximum number of cycles available during the history of the universe:
2×10^{10} years $\times$ 365 days $\times$ 24 hours $\times$ 60 minutes $\times$ 60 seconds $\times$ 2.9×10^{23} cycles $< 2 \times 10^{41}$ cycles total.

What is the largest belief set I could test?
Number of lines in truth table = 2^n,
where n = number of logically independent propositions.

2^{138} atoms require more than 3×10^{41} lines.

So, at 1 truth-table line checked/supercycle,
I cannot evaluate even the truth table for 138 independent propositions during the interval from the big bang to the present.

FIGURE 1. Costs of Truth-Functional Consistency Tests

[4] Similar questions have sometimes been raised in AI, with answers that agree fairly well with the estimates here. Minsky asserted that one million "elements of knowledge" would suffice for a machine with "very great intelligence" [42, p. 26]. Some medical AI workers have estimated a million facts (of only 10 Lisp words each) to be needed for internal medicine, and a million more for all the medical specialties [46]. In his production-system model of cognition in [3], Anderson asserts that an adult human being's "production memory" should contain between tens of thousands and tens of millions of production rules or procedures (there is also a large long-term declarative knowledge memory). Again, in an AI context, Lenat claims that, in their higher level analogical reasoning, people employ "perhaps a million distinct memories of objects, actions, emotions, situations, and so on" [36]. Lenat's current project is the construction of a large computer knowledge base of the real-world facts and heuristics contained, as commonsense knowledge, in a typical 30,000-article, one-volume desk encyclopedia (for comparison, Lenat claims the *Encyclopedia Britannica* contains nine times as many articles). He estimates this project will require about 300 programmer-years and one decade [37].

stance, the preface to the famous 1910 edition of the *Encyclopaedia Britannica* states the following:

> Whereas two or three centuries ago a single mind was able to acquire and retain in some one particular field of knowledge . . . nearly all that was worth knowing, and perhaps a good deal of what was best worth knowing in several other fields also, nobody now-a-days could, whatever his industry, cover more than a small part of any of those fields. [8]

Sixty years later the mathematician S. M. Ulam reported estimates that 200,000 theorems were published in recognized mathematical journals per year. He worried about the fact that "the number of theorems is larger than one can possibly survey . . ." and that "there is probably not one mathematician now living who can even understand all of what is written today" [56, pp. 287–291]. (Indeed, one wonders whether cognitive science itself has any better prospects of manageability.) In contrast to the Baconian ideal that "all learning is my province," or Peircean vistas of limitless acquisition of knowledge, our contemporary worldview seems to acknowledge the idea of an upper bound on the size of a human cognitive system.

TABLE I. Big Texts

Microcomputer operating system	≈5,000 lines of code
Webster's Collegiate Dictionary	50,000 entries
Oxford English Dictionary	500,000 entries
SDI software	10,000,000 lines of code
Cray supercomputer	10,000,000 pages of disk storage
Library of Congress	80,000,000 volumes
Human DNA	3,000,000,000 base units ≅ 1 billion bits (50,000 pp.)
Human cognitive/ perceptual system	≈10,000,000 items (?)

Let us first consider the relative size of some uncontroversial textual objects—books, libraries, software, and so on—despite the apples-and-oranges problem of commensurability of the diverse types of items involved (see Table I). With these quantities in mind, we can begin to grope toward some sort of crude numerological estimates of the approximate maximum size of the mind's putative program (perhaps within a couple of orders of magnitude). For the sake of argument, we assume some of the conventional cognitivist picture of the overall ground plan of a cognitive system as a convenient approximation.

The mind's dictionary for a normal adult's spoken vocabulary might range as large as an undergraduate's abridged *Webster's Collegiate*. (Although a 10,000-word active vocabulary is typical for a high-school graduate, a good reader can understand at least four times as many.[5]) We will also allow a "mentalese dictionary" of similar size for the internal language of thought (see Table II). We next include a corresponding mind's encyclopedia of the individual's basic factual information, commonsense knowledge, etc. This will be assigned as many entries as the two dictionaries combined, which is comparable to the size of actual encyclopedias. No mind could be complete without an additional personal history, an episodic memory of its own idiosyncratic microevents: Suppose one actually does "learn something new every minute." On average, the mind will gain one long-term memory in each 1-minute short-term memory cycle. (Haber reported that subjects' memory for over 2,500 pictures presented 1 every 10 seconds to be "essentially perfect" 1 hour afterward. Standing then showed that some subjects maintained 90 percent recognition accuracy days after viewing 10,000 pictures [28, 53]). In a 70-year life span (which equals 36,000,000 minutes), with one-third spent in dreamless sleep and nine-tenths of this stream of consciousness forgotten, over two million episodic memories could still accumulate. Finally, falling back on the old yin/yang of a fact/value distinction, I will add a goal structure that is of the same order of size as the mind's encyclopedia. (Procedural know-how, heuristic rules, etc., will just be subsumed within these declarative knowledge structures. Generally, the algorithm/data structure distinction will not be observed here; the debuggability predicaments of each are, for our purposes, similar.)

TABLE II. Estimating the Size of a Human Cognitive/Perceptual System

Mind's dictionary (words + concepts)	100,000
Mind's encyclopedia	100,000
Episodic memory	2,000,000[a]
Goal structure	100,000
Total cognitive elements	2,300,000
Visual font	200,000
Auditory, etc.	200,000
Motor schemata	200,000
Total perceptual/motor categories	600,000
Total: About 3,000,000 cognitive/perceptual/motor elements	

[a] A person's stream of consciousness:

Suppose "you learn something new every minute" (=STM);
70-year life span = 36,000,000 min.;
⅓ sleep (dreamless), and 9/10 forgotten;

Yields 2,000,000+ long-term episodic memories.

If we consider the perceptual abilities of a human being, the well-read person can learn 50,000 ideograms for written Mandarin Chinese (although 5,000 suffice as "basic Chinese" for reading a newspaper). Similarly, chess experts seem to work with a perceptual pattern set of about 30,000 board situations [50]. A Mandarin-reading chess player (perhaps even multilingual) who manages to find his or her way around town, recogniz-

[5] The average eight-year-old child can recognize over 14,000 root words; this works out, beginning at 18 months, to vocabulary growth at about 5 new words per day (see [10] and [41]).

Some convergent information on a cognitive/perceptual (C/P) system estimate:
Is there room in our heads for a 10 million–element system?

Total brain volume: ≈1.4 l.
Cortex volume: 0.5 l.
10^{10} neurons × ≈5,000 synapses/cell = 5×10^{13} synapses.

Suppose only 1/5 of the synapses are dedicated to representing C/P elements.
Then, 1 C/P element: 0.05 cu. mm volume; 1,000 neurons; 1,000,000 synapses.

Minimum information-representation capacity of cortical tissue:

Suppose each synapse ≅ 1 on/off bit.

1 C/P element ≅ 1 Mbit = 60 pp. of text.

Therefore, no Cartesian nonspatial substance is needed yet.
Versus, for example, if the C/P element/synapse ratio were worse than 1 : 24
(since 24 bits are required just to index 10 million elements).

FIGURE 2. The Volume of a Concept

ing the faces of acquaintances,[6] does not seem an unrealistic possibility. This individual's "visual pattern font" might reach 200,000 items. Yielding to visual chauvinism, I will assign a similar number for all other sense modalities combined. Finally, viewing motor skills on the same hierarchical model as visual pattern recognition, let us suppose there are as many motor schemata as visual patterns. (There will also be special-purpose modules for syntax and early vision, among others.)

The grand total for the inventory of a cognitive/perceptual/motor system is then around three million items. Even setting aside the conceptual morass of distinguishing where one item stops and another begins, we must still note that, on a standard cognitivist picture, these items will vary enormously in size. A proposition might be only about the size of the six-chunk capacity of human short-term memory, whereas a picture could be worth more than a thousand words. For the moment, we pass over the question of the average size of these items themselves.

Some rough neuroanatomical constraints can be used as a check on this size estimate for a cognitive system: For example, is there room in our heads for a 10 million–element system, or would crowding it in require a brain the size of a bathtub? (See Figure 2.) The cerebral cortex, generally regarded as the seat of higher cognitive functions in Homo sapiens and other animals that have them, has a volume of around 0.5 liter for human beings. (Keeping in mind that a cognitive/perceptual element might not be narrowly localized in the cortical sheet, the "volume of a concept" would then average 0.05 cu. mm of gray matter.) Although much information is represented in other features of brain structure, the synapse is predominantly viewed as the site of memory. The human cerebral cortex is now thought to contain 10 billion neurons, with perhaps about 5,000 synapses each; there are then about 5×10^{13} synapses total.[7] With this estimate of synaptic density, we can attempt a calculation of the minimum information representation capacity of cortical tissue—after all, it is not going to be unlimited.

Since there are other processing chores, not every single cortical synapse can be dedicated to representing cognitive/perceptual elements. If we suppose only a fifth of them are so occupied (indeed, under suitable conditions, individuals can lose one entire cerebral hemisphere without detectable deficit), there are then about one million synapses per cognitive/perceptual element. If each available synapse here is supposed to encode 1 binary bit of information, then a cognitive/perceptual element (including perhaps some of its interrelations to other elements) can be synaptically represented by 1 Mbit—a million bits—which would be about 60 typed, single-spaced pages of arbitrary alphanumeric text. All that this establishes is a convergent confirmation that the aforementioned cognitive system size estimate is not entirely unrealistic. No Cartesian nonspatial substance is needed yet.

In contrast, if the ratio of available synapses to cognitive/perceptual elements had turned out to be worse than 24 : 1, it would fall short of convergent confirmation of the system size estimate since 24 bits are required just for a minimal unique index of each of 10 million items. Or, again, a cognitive system the size of the 80 million–volume Library of Congress (with an average book of about 300 typed pages), or even of the 25 million–volume Lenin Library, would not be synaptically representable in a human cortex.[8]

[6] The right cerebral hemisphere seems to contain a face-recognition module with the capacity to learn 800 faces as easily as 90, and to reidentify them years later as accurately as a few months later. Carey and Diamond claim that "the limits on this enormous but everyday capacity, if any, have not yet been found" [11, p. 61].

[7] Accessible accounts that include the estimates above can be found in articles published in [6]. For some review and evaluation of quantitative neuroanatomy, see [15].

[8] A natural "informational origami" objection is that cognitive structure may well be represented in brain structure more subtly and efficiently than the crude additive manner sketched here. For example, it might be encoded with some of the compactness that brain structure information must be stored on the genome (see the "Prospects for Cognitivism" section). Sometimes, some of the DNA "Turing tape" even can palindromically serve multiple functions.

Mental structure, however, unlike lower order brain structure, seems unlikely to be amenable to much generative compactification. One reason derives from the familiar philosophical observation of the poor fit between the physical—here, the neuroanatomical—and the domain of everyday human situations and needs. The natural kinds of the physical/biological domain mesh only roughly relative to humanly relevant concepts and knowledge structures. (E.g., it is easy to envisage a compact genetic code for, say, a 1-to-1 mapping from retina to visual cortex, but distinctively more difficult to do so for a concept set including "telephone," "bachelor," etc.) Hence, exploitable regularities of psychological structure seem unlikely to correspond well with regularities of brain structure.

In addition, debate continues about whether human cognitive structure *is* regular in the first place, rather than an intrinsically arbitrary hodgepodge (see the "Mind As Kludge" section). Consider the huge variation in human belief systems that can be found today. In contrast, brain structure (e.g., cortical columns) seems a very large-scale but *repetitive* array. It is like the difference between geography and geometry, with geology somewhere in between. Consequently, the representation scheme for a cognitive system must be inefficient or inelegant to the extent that it must be flexible enough to accommodate the deeply ad hoc character of much human knowledge and interests.

UNVERIFIABLE SOFTWARE

If we took seriously the back-of-the-envelope estimate that a human cognitive system could have a size in the 1-to-10 million-item range, what would be some of the methodological consequences for cognitive science? One question this size estimate raises concerns the manageability or comprehensibility of the mind's program. Anatomists are fond of saying the human brain is the most complex physical structure known in the universe. Moving from the hardware to software level, one would expect the mind's program to be correspondingly complex. A comparison of the mind's program size estimate with the sizes of the big texts in Table I seems to confirm—with some exceptions—this qualitative difference in magnitude, unless the size estimate for the average individual cognitive/perceptual element is absurdly small, such as under one typed page.

In particular, if we compare the size of the mind's program with size estimates for the battle-management software for the Strategic Defense Initiative (SDI) (in the early "total-shield" concept, not more recent scaled-down proposals), the accepted estimates for the planned original SDI software ranged up to about 10 million lines of code (expected to be in the higher level computer language Ada). Roughly speaking, the size of the SDI battle-management software is then at least 100 times smaller than the estimated size of the mind's program. Is the proposed SDIO software envisaged to be comprehensible—in particular, "verifiable" or debuggable? This question has become a topic of current political controversy. In 1985 computer scientist David Parnas resigned from the SDI Panel on Computing in Support of Battle Management. His letter of resignation and an accompanying set of short tutorials arguing that the SDI software system would not be trustworthy were subsequently published over the ARPANET. Later, another paper with a similar conclusion, that reliable software for such a defense may be impossible, appeared as the lead "Science and Public Policy" article in *Scientific American*.[9] The basic idea that (moderately) large programming projects suffer problems different in kind from small ones had been vividly argued much earlier by Frederick Brooks, project manager for development of the IBM System/360 and its operating system, in [7]. The tenor of Brooks' essay is conveyed by the cover illustration, a paleontologist's drawing of the La Brea Tar Pits.

The Pascalian game theory of verification for the mind's program and for a ballistic missile defense software system differ vastly: The downside risk for a failure of a mind's program candidate is just a research embarrassment, whereas failure of SDI software could bring a catastrophe unprecedented in human history. It is also difficult to envisage a full-scale test of the SDI software system without global thermonuclear war, whereas this seems less of a problem for a mind's program candidate. Nevertheless, the basic verification predicament for each of these huge programs is essentially similar.

SOFTWARE VERIFICATION

What is program verification? In short, checking for local syntax errors in code is routine and computationally cheap; the trouble lies elsewhere. In software engineering, the conventional trichotomy of levels of design is (1) abstract specification of the software task or logical function, (2) devising the basic procedural idea or algorithm for the task, and (3) writing the actual detailed program code that implements the algorithm. Evaluation of the resulting software includes establishing correctness (for the appropriate range of inputs, the program yields only the right outputs), completeness (for the appropriate inputs, the program always yields an output), and computational tractability (for appropriate inputs, the computational costs of the program are always acceptable).

Real-world debugging tends to be perceived as only minor "noise" in the software development process; however, a number of studies indicate that the majority of the actual cost of a software project is devoted to such testing. Also, a range of studies suggest that, even when it is done properly, the resulting software will still contain around one bug for every 100 statements.[10] Indeed, bug corrections themselves tend to be significantly more error prone than the original code. Furthermore, apparently because of combinatorial explosion of transition possibilities, asymptotic bug rates often seem to go up, not down, with increasing program size (see, e.g., [7, pp. 88–89)].

Bugs seem to be endemic. The startling character of such statistics suggests that we tend to unrealistically presuppose that programs are essentially error free. Such a picture may itself be an instance of the recently much-studied pathological inability of human beings to estimate and reason in terms of accurate event base rates;[11] perhaps some vivid stereotype of "computer-like perfection" drives us to overlook observed bug frequencies. (A possible harbinger of a paradigm shift here is the ACM journal *Software Engineering Notes*, which regularly publishes lists of bug episodes.)

To an outsider, the classic verification methodology, exemplified by Dijkstra [19], seems to be self-consciously modeled on a Cartesian epistemology of deductive science: "Perfect" formal proofs of a program's correctness and completeness are required; they resemble metamathematical proofs of consistency and completeness of deductive systems. Although controversy surrounds this stringent program verification methodology, I think there are some reasonable questions about whether history's majority verdict will be that the full-scale methodology becomes more of an interesting art

[9] Parnas's material has since appeared in more extensive form as [45]. See also [38].

[10] These studies are reviewed in [4, pp. 1–3, 33–35]. See also [7, p. 20].

[11] A review of psychological studies of the role of human base-rate estimation pathologies in real-world risk assessment is found in [51].

for art's sake end rather than a real-world feasible technique. Parnas remarks that the programs for which he has seen such proofs have all been "well under 500 lines" [45, p. 439]. In the preface to a recent book in the Dijkstra tradition, Gries acknowledges that the methodology has been applied only to programs consisting of "one or two pages" of text. (Gries asserts that "finding an error should be the exception rather than the rule." Ironically, however, a typographical error occurs by the second page of his preface [27, pp. vii–viii].)

The underlying strategic problem for such perfectionism is a new wrinkle on the traditional epistemological problem of regress. Program verification, like theorem proving, tends to be computationally complex.[12] To establish adequate confidence in a program by this methodology, one moves out of the frying pan of questions about the program's correctness and into the fire of an even *more* real-world unmanageable correctness proof. A conventional proof of formal system consistency must always be relative—it must rely on some background level of intuitions about correctness. The classical concern is the adequacy of our metatheoretic notions of consistency that must be appealed to in such a consistency proof. For example, in 1936 Gerhardt Gentzen gave a consistency proof for elementary arithmetic using extremely powerful nonelementary methods (as required by Godel's Incompleteness theorem), that is, methods not formalizable in arithmetic itself involving transfinite induction. According to Kleene, the logician Alfred Tarski, "asked whether he felt more secure about classical mathematics from Gentzen's consistency proof, replied, 'Yes, by an epsilon'" [34, p. 257].

This type of no-win predicament is now mirrored at a practical level for program verification: When the proof itself must be so unmanageable, a Cartesian method of analyzing it down to a vast series of humanly obvious individual steps simply introduces its own unreliability. The very high observed error rates described earlier heighten this practical regress problem. There seems to be a point of diminishing returns: As programs and their verifications get bigger, the proofs' power as convincing machines thereby asymptotes. (I believe corresponding worries underlie much of the methodological puzzlement that greeted Appel and Haken's proof of the Four-Color theorem by computer-assisted exhaustive checking of huge numbers of possible cases.[13])

It is worth emphasizing that, although total correctness proofs may be an overreaction, there is a sound motive behind them. Dijkstra is concerned that, without such proofs, programmers live "in a limbo of folklore, . . . never quite sure what the system will do to their programs" [19, p. 202]. Indeed, in the beginning is our end: In "Computing Machinery and Intelligence," Turing already seemed to glimpse this predicament: "We also wish to allow the possibility that an engineer or team of engineers may construct a machine which works, but whose manner of operation cannot be satisfactorily described by its constructors because they have applied a method which is largely experimental [1, p. 7]."

A more applicable approach than perfect verification is probabilistic analysis of program performance. The basic idea is to make a representative sampling of a range of input instances from the total problem space and then to evaluate program behavior for those cases. The evaluation can be by formal methods, as in Smale's proof that computationally "hard" problem instances for the simplex linear programming algorithm are in a sense "rare" [52]; or the evaluation may be "empirical," by actually observing computer running times.[14] Thus, the goal shifts from "perfect" verification to establishing a sufficiently high probability of simply adequate program performance, as in engineering tests of conventional machinery. The crucial methodological puzzle in probabilistic program testing is, how is one to define the problem-space regions that are of interest (how much, how often) for a given purpose? Correspondingly, how is one to identify a sufficiently systematic sampling procedure on those problem-space regions? Problem instances are typically interesting or important only relative to a whole context of human goals, needs, and so on. Thus, probabilistic program verification becomes intimately and inextricably linked to exactly those volatile phenomena that are the most distinctively resistant to "clean, scientific" definition.

A PASCALIAN PREDICAMENT

It is important to be aware that some of the simplest software with the best track record of actual performance is quite buggy. The most startling stories I know concern floating-point arithmetic algorithms for microprocessors, for example, a square-root function that in some cases produces outputs differing from target values by an order of magnitude. William Kahan, of the Computer Science Department at the University of California, Berkeley, has argued that no calculator in the foreseeable future will be free of serious round-off errors.[15] Similarly, the operating system for the IBM PC microcomputer still harbors a number of nontrivial bugs; indeed, each new version seems to engender in turn its own new crop of them. (Thus, much of the business of user publications such as *PC Week* consists of bug patrolling and reporting.) Therefore, even micro-

[12] For a brief discussion, see [48]. A survey of practical and theoretical unfeasibility of formal proofs of program correctness is found in [39].

[13] See, for example, [55]. For an overview of another huge proof (15,000 pages long, primarily by hand, in group theory) see [26]. In arguing for his methods of probabilistic proof, Rabin has noted in effect (apparently not in print) the diminishing-returns problem for the convincingness of large mathematical proofs; see [35].

[14] For an empirical study of actual run times of a simplex package, see [40]. For an analysis of basic methodological issues involved in such empirical evaluations, see [24].

[15] See [30]. (I recall that Kahan's office used to be littered with pocket calculators that he could get to exhibit the most exotic behavior.) See also the discussion of analysis and control of errors in [29] (reportedly, it was largely ghostwritten by Kahan). The approach there is toward learning to live with unavoidable errors as design limitations rather than as unanticipated bugs—that is, to predict and manage important classes of them.

processor software engineering seems to diverge from the unclouded optimism of any prospect of monotonic refinement toward the ideal (e.g., along the classical pragmatist lines of C. S. Peirce's model of scientific inquiry). The decision-theoretically correct conclusion here is not some Luddite advice to trash our calculators; the rather unsystematic intuition, on the basis of which we are willing to design bridges, is that we can be reasonably certain that the buggy cases are quarantined or identifiable when they do get loose. The line between bug and undocumented design feature can blur; we learn to live sometimes with bugs along the lines—"A weed is just a plant for which we have not yet found a use." We do not demand perfection. We thereby bet our lives on hunches about the probability distribution of the bad cases; but the alternative is a kind of Cartesian paralysis.

It is important to be aware that some of the simplest software with the best track record of actual performance is quite buggy.

When one scales up the software to programs recognizable as AI, intuitions about proper risk assessment become more hesitant. For example, a number of medical diagnosis expert systems are among the oldest and most impressive successes in AI. MYCIN, an expert system for the diagnosis of infections, has been outperforming human physicians for over a decade. Yet Lawrence Kingsland, head of the expert-systems project at the National Library of Medicine, has reported that to his knowledge only a very small handful of diagnostic medical expert-system programs (which did not include MYCIN) were yet being used at all in regular clinical practice. (Indeed, the Federal Drug Administration has been reviewing the question of certification of medical diagnostic systems.)[16] In this context, the manufacturer's standard denial for a mere word-processing program of even implied warrantability of fitness for use of any kind (it is always provided on an "as-is" basis, and so on) would be unheard of for a conventional machine such as an automotive brake system. In fact, the laws of some states do not permit such exclusion of implied warranty. These remarkable disclaimers now take on the character of a further symptom of the practical undebuggability of even small-sized software.

Thus, to travesty Verlaine's aphorism on the undebuggability of a poem ("A poem is never completed, only abandoned"), it seems that "a program is never finished, only released." Large software will be condemned to the Sisyphean regime of the iron law of Murphy. When we extrapolate beyond actual software to the complete purported program of the human mind, the cost-benefit analysis becomes still more problematic. Of course, actual human reasoning procedures diverge far from formal perfection. As reviewed in the upcoming discussion of quick-and-dirty heuristics, their unreliability seems a profoundly Faustian bargain, indispensable to avoid computational paralysis. But the peculiar possibility emerges that, even if a fully intelligent program could be constructed, it might be rightfully condemned to remain a toy or laboratory curiosity. Could we rationally rely on it as much as we depend on a human taxi driver, for instance?

The corresponding possibility from the perspective of the cognitive science millennium would be that the Martians might present us with the correct vast program for the human mind; or, more plausibly, Mother Nature might blindly have constructed it over the aeons. But we might be unable to verify or evaluate it in the sense explained, much less to establish that the program succeeds in mapping the human mind (including bug-isomorphism). The point is that this unmanageability might stem from the essential nature of the human mind, from the intrinsic structure of the mind's program—its vast size, for example—rather than from mere historical accidents of slovenly programming style.[17] Thus, an appearance/reality distinction: A huge program for the mind may be objectively possible, but it may also be that we cannot establish its existence. The mind's program may then be a *Ding an sich* that is practically unknowable to a philosophically interesting extent.[18] Of course, complete understanding of the mind's program is still possible in principle, as is the use of hopelessly unwieldy quantum mechanics instead of classical mechanics in designing a bridge. But the whole thrust of a resource-realistic philosophical perspective is just that so abstract a feasibility is cold comfort.

THE TEXTURE OF FAILURE

Software for a cognitive system will differ in its failure proneness from conventional types of machines for reasons other than just its brute complexity. A cognitive system's program will tend to act as a failure amplifier because of its intrinsically branchy structure and its distinctively holistic structure of interconnection. Consider the vivid contrast between reliability of computer software and hardware. Running any given program requires a machine that is more complex than the software, in the most primitive sense of the number of comparable elements. First, the hardware must include

[16] For some performance evaluations of MYCIN, see [9, part 10]. (A caution: Reports of software outperforming human beings on statistical prediction tasks must be taken *cum grano salis*. Given the startling ineptness of people at reasoning in terms of objective base rates as noted earlier, it is no surprise that the crudest actuarial formulas outperform clinical experts. For a review of the three decades of studies of these comparisons, see [44, chap. 7] (especially pp. 139–141).) Kingsland's remarks are from [33].

[17] Only the latter, contingent type of defect seems to be the source of the program incomprehensibility discussed by Weizenbaum in [57, chap. 9].

[18] I have discussed this concept, for example, in [13, chap. 6].

at least one independent memory location for storing each symbol of the program. Yet commercial software continues to have defect rates that would be exotic for conventional machinery, current computer hardware surpasses any other artifact of our era in both its intricacy and reliability. Hardware failures are relatively rare events. Why is this? The remarkable record of hardware reliability owes something to a concern with such things as fault-tolerant design and error-correcting codes that dates back to von Neumann.

But the difference between hardware and software that is mainly responsible for this qualitatively different order of failure proneness concerns structure. Tacitly we tend to extend our picture of the behavior of conventional artifacts to software, but the latter is in fact an entirely different regime. A massive memory (e.g., a disk system) differs from a massive program it contains in that the memory has a simple additive structure, rather than a branching complexity. Very roughly, a satisfactory memory medium (e.g., basically iron filings embedded in Mylar sheet) just has to be tested for purity and uniformity to ensure there are no imperfections in the physical material. Unlike software, the diskettes that contain it are routinely guaranteed free of defects in materials and workmanship. To test a memory to establish that it is good, one only has to sample and check the relatively independent elements. The failure modes of memory media do not include cases where insidious, fine-structured, extensive propagation is likely.

The texture of failure for a program differs because of its branchy structure: There are not just many distinct states, but a combinatorially exploding number of ways in which one state can be connected to another. The branching is not just wide, but deeply hierarchical in organization, nested iteratively through many layers as in truth tables. Debuggability by exhaustive checking of this vast number of potentially possible transitions is then not remotely feasible. (Recall the example of the costs of truth-functional consistency testing sketched earlier.) Here again we confront a problem of scale.

In addition, the essential structure of a cognitive system ensures that computational approximations of it will function as failure amplifiers. In philosophy, Quine [47, sec. 6] and Davidson [18] have long emphasized the distinctively holistic character of cognitive systems. But that interconnectedness means that defects, as well as revisions, will tend to propagate via the flow of information throughout the web of belief. In this way, such a nexus acts as a bug detector, rather like a spiderweb. Divide-and-conquer software design methodologies that prescribe a hierarchy of self-contained, clean-interfacing modules and submodules are, of course, good strategy; but the intrinsically holistic nature of program models of cognition entails limits to such modularity. Quine, Davidson, and the philosophical tradition they epitomize deny or recoil from departures of actual cognitive webs from ideal rationality; it is therefore ironic that such nonidealities—nonclosure of the belief set under deduction, compartmentalization of it, and so on[19]—act as a type of fault-tolerant software design feature, namely, as valuable quarantines on bug propagation. (So that, e.g., contrary to Quine and others, a contradiction in the system does not in fact actually threaten to generate every proposition.)

MIND AS KLUDGE

Another source of undebuggability is intrinsically messy program structure. Over the past two decades, a set of methodological constraints on the very form of a cognitive science have been urged, particularly by the Cambridge Cartesians—Chomsky, Fodor, et al. (see, e.g., [16, chap. 1] and [22]). The correct form of a cognitive theory must be as a small set of elegant, powerful, general principles, on a model such as classical mechanics, because such hyper-Euclideanism is supposed to be part of the essence of good scientific explanation, just as being a male sibling is the nature of brotherhood. It has been claimed that, if the mind turns out not to have this clean, compact structure, but instead turns out to be designed as a patchwork hodgepodge of many ad hoc kludges (i.e., inelegant rules), then there can be no cognitive science. Whatever cognitive scientists are doing, it will be merely something low level and descriptive, like geography or writing large, messy Russian novels.

It is at least a question of current empirical investigation, and of some controversy, whether the mind is in fact hyper-Euclidean in this way. After all, what is elegant to the eye of the cognitive theoretician may not be obediently implemented by Mother Nature because it may not be efficient for actual processing: Thus, the alternative emerging picture of mind instead as an in-practice anomalous kludge of limited-applicability special-purpose procedures. But how much comfort can we take from a kludge model of the mind?

If all of the mind had a neat, regular (and modular) hyper-Euclidean structure, its software would be relatively easy to debug on the pattern of conventional consistency and completeness proofs for formal systems in logic as mentioned earlier. But kludge-structured software for the mind's vastness will be seriously harder to verify just because it's messy. In a backhand way, for an unanticipated reason, the Cambridge Cartesian commitment to hyper-Euclideanism is at least somewhat vindicated. Whether or not a science of the kludge ceases in essence to be *science*, it is in danger of being impossible, because a huge kludge threatens to be a practically unknowable entity.

THE QUICK AND THE DIRTY

It is important to distinguish between a kludge and a quick-and-dirty procedure—one that is computationally cheap, but formally incorrect and incomplete. A program can be inelegant, but still correct and complete. Conversely, a program can be compact, but incor-

[19] I have discussed these in [13, chaps. 3–4].

rect and/or incomplete. A variety of converging evidence has emerged that suggests for a wide range of reasoning procedures a profound antagonism between formal correctness/completeness and computational tractability (see, e.g., [13, chap. 4]). At least the flavor of these complexity-theoretic results is conveyed by the example of the computational cost of tautology testing by truth tables, mentioned at the onset of this article. The conclusion I have drawn, which I think has independently become a leitmotiv of a variety of current research, is that human beings must evade this computational paralysis by using quick-and-dirty procedures.

The optimism of this picture of quick-and-dirty heuristics as the ultimate speed-reliability trade-off, however, is darkened when we turn to evaluation of the performance of such heuristics. Perfection may be impossible, but it is simple. It is no accident that correct and complete procedures were objects of study long before quick-and-dirty heuristics. For example, the concepts of perfect correctness and completeness for a deductive system are rather easy to define, and it is often relatively easy to establish that a system satisfies them. A quick-and-dirty deductive system is, of course, guaranteed not to be correct and complete; the most that can be sought is that it possess *acceptable* incorrectness and/or incompleteness. But notions of such acceptable "dirtiness" will tend strongly to have the character that we already saw earlier for probabilistic software verification.

That is, to evaluate whether a program is acceptably dirty, one has to be able to specify, again, the subset of input problem instances that are of interest—for example, that are likely to arise often in relevant conditions. Blind trial and error cannot suffice. Then one must estimate whether the program's behavior will be correct, complete, and computationally tractable sufficiently often in those cases. Such probabilistic analyses of heuristics are just beginning to emerge, principally in the areas of computer science that deal with optimization procedures that are computationally costly and much more clearly defined than most AI tasks.[20]

One leading worker in the area, Richard Karp, views the venture of probabilistic analysis of combinatorial optimization algorithms as "only partially successful" because of continued use of "the most simplistic of probabilistic models" [31, p. 108]. He adds later, "There is a really fundamental methodological problem: How do you choose the probability distributions? How can you possibly know what the population of problem instances is going to be?" [23, p. 113]. Indeed, Karp and Luby felt obliged to conclude a recent paper proposing and formally analyzing some Monte Carlo techniques by citing results of actually running the algorithm on six examples [32, p. 63]. In-practice performance evaluation still seems to compel a quasi-experimental approach. A basic difficulty, once again, seems to be that complete specification of the set of relevant typical problem instances will often be delicately enmeshed in the whole context-sensitive nexus of human goals, needs, and purposes. Such probabilistic analyses are not merely hard to accomplish as a practical matter; the very concept of *success* here does not seem susceptible of clear specification. Therefore, the threat is of the reliability of the heuristic becoming merely a haphazard "so far, so good" black-box gamble.

[20] For an overview of such probabilistic analyses, see Karp's ACM Turing Award Lecture [31] (pp. 106–108 are especially relevant here) and also the interview with Karp in the same issue [23].

A familiar intuition regarding the complexity of the human brain is that a low-level description of its structure would be entirely unmanageable for us.

It is in this way that a higher order dilemma arises between computational tractability and software verification: Quick-and-dirty procedures do indeed seem indispensable for evading intractability, but at the cost of loss of clearly definable concepts of verification or debugging. (In philosophy, the prospect of mind as inevitably quick-and-dirty kludge poses methodological puzzles in this way for recent reliabilist theories of justification such as Goldman's [25], which seeks to evaluate basic cognitive processes epistemologically in terms of their reliability, power, and speed.) In still another way, the mind's software seems to recede as a practically unmanageable *Ding an sich*.

PROSPECTS FOR COGNITIVISM

A familiar intuition regarding the complexity of the human brain is that a low-level description of its structure would be entirely unmanageable for us. For example, "Even if we had a diagram that included every one of the billions of neurons and billions of interconnections in the human brain, it would stare at us as mutely as the grains of sand in a desert" [5]. As Anderson asserts, "A neural explanation is too complex and detailed to adequately describe sophisticated human behavior. We need a level of analysis that is more abstract" [2, p. 11]. The natural inference has then been to the promise of a computational psychology. The most powerful level of description of the brain should be at the program level of abstraction, rather than, say, a microlevel neuroanatomical wiring diagram.

The argument here, however, has been that such a program seems likely still to be unmanageable—unevaluatable, in particular—because of its vastness and its branchy, irregular, and necessarily quick-and-dirty structure. This picture of mind as huge kludge in turn seems to account etiologically for some of the current practice, as opposed to ideology, of AI. The conventional optimist view of AI methodology is that it imposes valuable discipline on cognitive modeling. First, it serves a Socratic function, forcing us to be honest by focusing attention on specific implementation details

of the model. We thereby find and confront the "and-then-a-miracle-happens" gaps. Second, AI programming—with performance evaluation—constitutes a kind of quasi-empirical test of the model. The depressive critique, however, familiar since Dreyfus's *What Computers Can't Do* [20], is that AI's approach to *problem-stellung* has in fact involved selection of "toy," simplified problem instances and quite limited problem domains, with attendant difficulties of "brittleness" or nonextendability of the resulting custom-tailored initial program designs. First steps abound, but few second steps.

We can now see that such practice is not some mysterious infantilization or moral defect. It can be made sense of as an instance of the inattention to scale that I proposed earlier is a pervasive feature of methodology throughout cognitive science. Difficulties arise because a human cognitive system will be so huge that the distance between pilot and full implementation will tend to be quite vast. Similarly, the inattention to systematic performance evaluation of AI programs seems more than historical accident. Rather, it can be viewed as a symptom of simple but deep features of the structure of a cognitive system. Mind as huge kludge poses specific obstacles to debuggability: identifying the problem space and the subregions that are of real interest, devising sampling strategies for them, and verifying that performance reaches reasonable levels. Recent neurally inspired connectionist[21] conceptions of distributed and massively parallel architectures can only exacerbate these structural unevaluatability difficulties.

As a concluding irony, it is interesting to turn our attention once more to scale to compare the size estimate for the mind's program with the information content of a human DNA strand (see Table I). The brain's bottom-level neuroanatomical characterization cannot be too huge, for it must be able to pass through the *genomic bottleneck* of DNA representability: The brain's blueprint must be able to fit into less than three billion base units, the single-copy size of the total human genome. With allowances for informational inefficiencies of protein specification, repeated sequences, noncoding introns, and so forth, somewhere between 10,000 and 30,000 typed alphanumeric pages of brain-specific genetic "text" seem available for representing the mind's hardware.[22] Whatever the vagaries of comparing lines of SDI code, cognitive items, and DNA text, it then seems that a human brain's genetic blueprint is orders of magnitude smaller than the proposed SDI software, much less the mind's program. The initial intuition about complexity of brain structure vastly exceeding that of the mind's program then turns topsy-turvy. The paradoxical conclusion is that a quite low-level description—the DNA representation—of the neural hardware turns out to be demonstrably more economical than the envisaged mind's software of cognitivism.

CONCLUSION

My basic point has been simply that there are reasons to expect the mind's program to be inhumanly unmanageable stemming from a resource-realistic approach to cognitive science. To this extent its usefulness to cognitive science itself will be limited. The distinctive recalcitrance of a science of the mind, compared with the physical and biological sciences, has long been noted. We can now see that, correspondingly, the mind's program would differ fundamentally from conventional machines. The mind's program would be an impossibility engine, in that it would be practically unfeasible for us fully to comprehend and evaluate it. This is not quite Edmund Husserl's verdict that articulating the structure of mind is "an infinite task"; but then bare in-principle possibility seems of limited interest. Nor is the point an impossibility proof of a complete mind's program cannot be constructed; rather it is that we cannot deal with or fully understand it. Our stance toward it might end up a little like that of coral animals toward the vast reef they have built.[23] (Recall Ulam's worries quoted earlier about a similar current state of mathematical knowledge.)

This argument does not impugn the plausibility of manageability of all very large programs, only ones with the distinctively unwieldy structure of models of the human mind. The present discussion also leaves open the possibility that there might be some program that could yield the full range of human-level intelligent behavior, yet have a radically different, in particular, simpler, structure than that of the human mind. (This possibility is explored in [14].)

Even with this in-practice anomalousness, some of the overall ground plan of the mind's program should be humanly accessible. Also, the smaller a submodule (when cleanly isolable), the more manageable it should be. I therefore feel that it would be an overreaction here to leap to a grand eliminativist conclusion that computational or cognitive levels of explanations are fundamentally misbegotten. (E.g., to conclude that such categories cannot be the natural kinds for a successful science of the mind.) Why should we expect a preestablished harmony, where human nature must turn out to be conveniently comprehensible to us?

I tend to personally lean toward a mutual coexistence: With a picture of the mind's program as huge, branchy, quick-and-dirty kludge, the conventional contrast in manageability cited earlier between explanations at the level of abstract program and of neural hardware fades. Full-scale software description of the mind faces a predicament of diminishing returns. A 10 million-element cognitive system with each element 100 statements long, and a 10 billion-neuron cerebral cortex begin to some degree to converge in complexity. Hence, the idea of the mind's program as an impossibil-

[21] For an introduction, see [49] and the articles in [17].

[22] See the papers in [43]; for an accessible review of the molecular biology, see [54].

[23] An emerging symptom of this phenomenon that I lampooned in a recent cognitive science fiction piece [14] is AI engineering steadily outpacing theory; that is, increasing practical AI successes with diminishing understanding of how and where the software works.

ity engine suggests some redefinition of research agenda, namely, that not too much is to be lost by turning back to neuroanatomy and neurophysiology. In fact, this has been a recent trend in cognitive science; the field seems to have recalled lately that cognition is, after all, accomplished with a brain. In this article another, perhaps once unperceived motivation for this tendency can now be discerned.

Acknowledgments. For generous help on this article, I am indebted to William Gasarch, Alvin Goldman, Mark Tuttle, and John Waclawsky.

REFERENCES

1. Anderson, A., Ed. *Minds and Machines.* Prentice-Hall, Englewood Cliffs, N.J., 1964.
2. Anderson, J. *Cognitive Psychology and Its Implications.* Freeman, San Francisco, Calif., 1980.
3. Anderson, J. *The Architecture of Cognition.* Harvard University Press, Cambridge, Mass., 1983.
4. Beizer, B. *Software Testing Techniques.* Van Nostrand Reinhold, New York, 1983.
5. Bernstein, J. A.I. *New Yorker* (Dec. 14, 1981), 121.
6. The brain, *Scientific American 241* (1979).
7. Brooks, F. *The Mythical Man-Month.* Addison-Wesley, Reading, Mass., 1975.
8. Bryce, J. Prefatory note. In *Encyclopedia Britannica.* Vol. 1, 11th ed. Encyclopedia Britannica Co., New York, 1910, pp. vii–viii.
9. Buchanan, B., and Shortliffe, E., Eds. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Mass., 1984.
10. Carey, S. The child as word learner. In *Linguistic Theory and Psychological Reality,* M. Halle, J. Bresnan, and G. Miller, Eds. MIT Press, Cambridge, Mass., 1978.
11. Carey, S., and Diamond, R. Maturational determination of the developing course of face encoding. In *Biological Studies of Mental Processes,* D. Caplan, Ed. MIT Press, Cambridge, Mass., 1980.
12. Charniak, E., and McDermott, D. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, Mass., 1985.
13. Cherniak, C. *Minimal Rationality.* MIT Press, Cambridge, Mass., 1986.
14. Cherniak, C. The wager. *AI Mag. 7* (1986), 120–124.
15. Cherniak, C. Meta-neuroanatomy: The myth of the unbounded mind/brain. To be published.
16. Chomsky, N. *Reflections on Language.* Pantheon, New York, 1975.
17. *Cognitive Science 9* (1985).
18. Davidson, D. Psychology as philosophy. In *Essays on Actions and Events.* Oxford University Press, New York, 1980.
19. Dijkstra, E.W. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, N.J., 1976.
20. Dreyfus, H. *What Computers Can't Do.* 2nd ed. Harper and Row, New York, 1979.
21. Fodor, J. *The Language of Thought.* Crowell, New York, 1975.
22. Fodor, J. *The Modularity of Mind.* MIT Press, Cambridge, Mass., 1983.
23. Frenkel, K.A. Complexity and parallel processing: An interview with Richard M. Karp. *Commun. ACM 29,* 2 (Feb. 1986), 112–117.
24. Golden, B., Wasil, E., and Baker, E. Experimentation in optimization. *Eur. J. Oper. Res. 27* (1986), 1–16.
25. Goldman, A. *Epistemology and Cognition.* Harvard University Press, Cambridge, Mass., 1986.
26. Gorenstein, D. The enormous theorem. *Sci. Am. 253* (1985), 104–115.
27. Gries, D. *The Science of Programming.* Springer-Verlag, New York, 1981.
28. Haber, R. How we remember what we see. *Sci. Am. 222* (1970), 104–112.
29. Hewlett-Packard. Accuracy of numerical calculations. In *HP-15 C Advanced Functions Handbook.* Hewlett-Packard, Cupertino, Calif., 1982, appendix.
30. Kahan, W. Mathematics written in sand. In *Proceedings of the Joint -Statistical Association Meeting* (Toronto, Ontario). 1983.
31. Karp, R.M. Combinatorics, complexity, and randomness. *Commun. ACM 29,* 2 (Feb. 1986), 98–111.
32. Karp, R.M., and Luby, M. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *J. Complexity 1* (1985), 45–64.
33. Kingsland, L. Overview of medical expert systems. In *Proceedings of the American Society for Information Science Meeting, Potomac Chapter* (College Park, Md., Nov.). ASIS, Washington, D.C., 1985.
34. Kleene, S. *Mathematical Logic.* Wiley, New York, 1967.
35. Kolata, G. Mathematical proofs: The genesis of reasonable doubt. *Science 192* (1976), 989–990.
36. Lenat, D. Computer software for intelligent systems. *Sci. Am. 251* (1984), 204–213.
37. Lenat, D., Prakash, M., and Shepherd, M. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Mag. 6* (1986), 65–85.
38. Lin, H. The development of software for ballistic-missile defense. *Sci. Am. 253* (1985), 46–53.
39. Manna, Z., and Waldinger, R. The logic of computer programming. *IEEE Trans. Softw. Eng. SE-4* (1978), 199–229.
40. McCall, E.H. Performance results of the simplex algorithm for a set of real-world linear programming models. *Commun. ACM 25,* 3 (Mar. 1982), 207–212.
41. Miller, G., and Gildea, P. How children learn words. *Sci. Am. 257* (1987), 94–99.
42. Minsky, M., Ed. *Semantic Information Processing.* MIT Press, Cambridge, Mass., 1968.
43. Molecular neurobiology. In *Proceedings of the Cold Spring Harbor Symposia on Quantitative Biology, 48* (Cold Spring Harbor, N.Y.). 1983.
44. Nisbett, R., and Ross, L. *Human Inference.* Prentice-Hall, Englewood Cliffs, N.J., 1980.
45. Parnas, D. Software aspects of strategic defense systems. *Am. Sci. 73* (1985), 432–440.
46. Pauker, S., Gorry, G., Cassirer, J., and Schwarz, W. Toward the simulation of clinical cognition: Taking a present illness by computer. *Am. J. Med. 60* (1976), 981–996.
47. Quine, W. Two dogmas of empiricism. In *From a Logical Point of View.* Harvard University Press, Cambridge, Mass., 1980.
48. Rabin, M. Theoretical impediments to artificial intelligence. In *Information Processing 74,* J. Rosenfeld, Ed. North-Holland, Amsterdam, 1974.
49. Rumelhart, D., and McClelland, J. *Parallel Distributed Processing.* Vols. 1–2. MIT Press, Cambridge, Mass., 1986.
50. Simon, H., and Gilmartin, K. A simulation of memory for chess positions. *Cognitive Psychol. 5* (1973), 29–35.
51. Slovic, P., Fischhoff, B., and Lichtenstein, S. Facts versus fears: Understanding perceived risk. In *Judgment under Uncertainty: Heuristics and Biases,* D. Kahneman, P. Slovic, and A. Tversky, Eds. Cambridge University Press, New York, 1982.
52. Smale, S. On the average number of steps of the simplex method of linear programming. *Math. Program. 27* (1983), 241–262.
53. Standing, L. Learning 10,000 pictures. *Q. J. Exp. Psychol.* 25 (1973), 207–222.
54. Sutcliffe, G. mRNA in the mammalian central nervous system. *Annu. Rev. Neurosci. 11* (1988), 157–198.
55. Tymoczko, T. The four-color problem and its philosophical significance. *J. Philos. 76* (1979), 57–83.
56. Ulam, S.M. *Adventures of a Mathematician.* Scribner's, New York, 1976.
57. Weizenbaum, J. *Computer Power and Human Reason.* Freeman, San Francisco, Calif., 1976.

CR Categories and Subject Descriptors: I.2.0 **[Artificial Intelligence]**: General
General Terms: Performance, Verification
Additional Key Words and Phrases: Cognitive science, mind/brain science, philosophy of mind

Author's Present Address: Christopher Cherniak, Department of Philosophy. The University of Maryland, College Park, MD 20742.