

ARTICLE***Liability For Distributed Artificial Intelligences******Curtis E.A. Karnow*** †**TABLE OF CONTENTS****I. INTRODUCTION**

- A. Complex Digital Systems
- B. The Future of Intelligent Systems

II. THE DEVELOPING TECHNOLOGY

- A. Looking Back: The Classic Expert System
- B. Looking Forward: Fluid Systems
- C. The Unreliability of Software
- D. Multi-Agent Networked Collaboration
- E. Agents and Network Viruses
- F. Polymorphism and the Units of Programming Action
- G. Summary Forecast

III. CAUSATION IN THE LEGAL DOMAIN

- A. The Conundrum of Causation
- B. The Tort System's Causation Analysis

IV. CAUSATION IN THE DIGITAL DOMAIN

- A. An Example of An Intelligent Processing Environment
- B. Unpredictable Pathology: Absolving Humans of Liability
- C. Failure of Causation: Absolving Programs of Liability

V. TURING, WHERE ANGELS FEAR TO TREAD

- A. The Registry
- B. A Comparison to Traditional Insurance

VI. PERSPECTIVES

[W]hat is interesting here is that the program does have the potential to arrive at very strange answers . . . ¹

Even the most intelligent among machines are just what they are—except, perhaps, when accidents or failures occur Could it be that artificial intelligence, by manifesting this viral pathology, is engaging in self-parody—and thus acceding to some sort of genuine intelligence?²

I. Introduction

Developments in that branch of computer science known as "artificial intelligence" (AI) have passed beyond the boundaries of the laboratory; early samples are now in widespread commercial circulation.³ At the same time, important new directions in AI research imply that the last decade or so of disappointing results will give way to truly creative, arguably intelligent, programs.⁴

Taken together, these two developments strongly suggest that in the next decade AI will be able to supply genuinely useful decision-making programs which operate in the real world and make decisions unforeseen by humans. This article forecasts the behavior of these intelligent programs and argues that they will inevitably cause damage or injury. The article goes on to suggest that, in the context of litigation stemming from such damage, insuperable difficulties are posed by the traditional tort system's reliance on the essential element of causation. Following the author's preference for technological answers to technological problems, this article concludes by offering not a new legal model, but a technological mechanism to compensate those injured by the unpredictable acts of artificial intelligences. This proposal, termed the "Turing Registry," is distinct from usual insurance schemes. In contrast to the traditional tort system, the Turing Registry proposal does not depend for its efficacy on establishing a proximate cause relationship between the injury and the agent of machine intelligence. This solution pretermits the issue of whether a specific intelligent program is "responsible" for the injury.

A. Complex Digital Systems

This article assumes that we will indeed employ artificial intelligence throughout the economy, for our infrastructure is increasingly found in complex digital systems. Business, the professions and personal work increasingly depend on the processing of computer data. The power of the digital machine allows an exponential increase in complexity, which in turn requires increasing computer power and in any event makes it impossible to turn back to manual processing.⁵

The notion of "complexity" is elusive, having technical meanings in a variety of disciplines.⁶ As used here, "complexity" connotes multiple interacting but independent elements. For example, a society may be thought of as a combination of interacting but independent persons. The sum behavior is a function of interactions with one's fellows as well as many individual characteristics. A car, and even more so an airliner, qualify as complex systems. As complexity increases it becomes difficult, and sometimes impossible, to predict the sum state of the complex system.

Complex computing environments are a function of the number of linked processing elements (both hardware and software) and persons (users and programmers). Plainly, this type of complexity is escalating. For example, the number of small controller computing chips (selling in the range of \$2 through \$5) is likely to increase dramatically over the next decade. In 1970, the typical car, office and home had no such chips integrated into its systems. About 100 such chips per unit resided in homes, cars and offices in 1990, and about 300 chips per unit are forecast for the year 2000.⁷ This is a part of a trend known as "ubiquitous computing," intensively studied by Xerox PARC in Palo Alto.⁸

These developments echo the spread of personal computers (PCs), which since 1981 show a total rise from zero in 1981 to about 100 million in 1989 to about 180 million in 1993.⁹ The number of transistors on Intel chips has risen exponentially from about 10,500 transistors on the 8088 chip through one million transistors on the '386 chip to the projected 100,500,000 transistors of the P7 chip.¹⁰ The number of lines of code in operating systems for personal computers has risen from less than 100,000 in early versions of DOS (about fifteen years ago), to three million lines in Windows 3.1, to roughly ten million lines of code in the current Windows 95.¹¹

These somewhat arbitrary measures do not begin to capture the complexity at issue. Far more importantly, the number of domains of human endeavor taking place in the digital context has rapidly increased as well. It is now trivial to note that commerce, from advertising to banking and credit transactions, flows in a digital environment. Entertainment is both created and provided electronically, and social interactions—from war to art to intimate associations—are increasingly electronically mediated.¹²

The movement to networked systems enables this process.¹³ Our computer systems do not stop at the box on our desks; they reach and meld with other systems around the world. Both data storage and data processing occur in those multiple remote locations, the congregation of which some call cyberspace. The number of on-line service subscribers worldwide is forecast to rise from ten million in 1990 to about fifty-five million in 1998.¹⁴ Host computers on the Internet have been estimated to be multiplying at the rate of nine to twelve percent *per month*.¹⁵ Increasingly, corporations are accessing *distributed* data. These systems manage information residing on physically-separated machines, although to the user everything appears to be local.¹⁶

That expansion of networked power comes with a price: the magnification of complexity. We use a multiplicity of data formats, operating systems and applications. We employ a variety of communications protocols, not to mention the infinite combinations of hardware. And we produce, maintain, modify and feed into these linked systems a practically unlimited amount of data.¹⁷

The inextricable complexity of essential digital equipment can make life miserable for the humans who are expected to operate the systems. These human operators will not reject the help of imaginative and creative programs that seem to know their way around the electrosphere. Such programs, currently used to search and filter information from the Internet, have been dubbed "intelligent agents." Intelligent-agent technology "is going to be the only way to search the Internet, because no matter how much better the Internet may be organized, it can't keep pace with the growth in information."¹⁸

B. The Future of Intelligent Systems

The programs that promise to be most useful are not the classic "expert" systems¹⁹ that mechanically apply a series of rules to well-defined fact patterns. To be sure, those expert systems are particularly useful in three contexts: first, where the system embodies years of human experience that otherwise might not get collected or analyzed;²⁰ second, where speed of operation is essential, such as an emergency nuclear reactor shut-down procedure;²¹ and third, where it is cheaper to use unskilled labor to implement an expert's recorded knowledge than it is to hire the human expert.²²

But far more useful will be autonomous "intelligent agents." The AI programs of interest, the successors to today's intelligent agents, will collect information without an express instruction to do so, select information from the universe of available data without direction, make calculations without being told to do so, make recommendations without being asked and implement decisions without further authorization.²³

A simple example of such a system is an airline reservation program.²⁴ The system would have access to a user's phone calls made through a computer program, the user's travel itinerary (available on a computerized calendar) and other computerized information. The system would note the correspondence between various calls to people in the 213 area code and trips planned for the following Friday. The program would call into United Airlines' computerized reservation system and book an appropriate flight and perhaps a hotel room for the user.²⁵ Details of a more complex, if hypothetical, intelligent system are provided below.²⁶

Future intelligent programs will not be cosmetic, entertaining flourishes on the front of dumb applications. Rather, the programs will truly execute their decisions with real data in a complex networked environment, and will affect real world events. We already have the forerunners in mundane operation. The New York Stock Exchange, large passenger airliners such as the Airbus, the telephone²⁷ and electric grids, and other computerized applications are all entrusted with decision-making authority affecting real money and real people. Some of these systems have already manifested unexpected behavior.²⁸ That behavior is considered an aberration; when it happens steps are taken to correct the aberration and eliminate the unexpected actions of the software.

The legal system thinks it knows how to handle unpredictable systems. When mistakes are made, one simply traces back the vector of causation to the negligent human agency that caused the error. Then, in theory, one sues that agency for damages and is made whole. The sins of omission and of commission are just as subject to legal condemnation as negligence, recklessness, intentional malfeasance or other human culpability.

However, some systems may be designed to be unpredictable. In addition, some decision-making programs will be highly distributed.²⁹ In these cases, results will be derived from a large number of concurrently-interacting components, from a wide range of

sources, machine and human, none alone able to make or manifest the "error." "Fixing" these unpredictable systems to operate predictably will eviscerate and render them useless. Programs flexible enough to be delegated our judgments must, of necessity, be expected to err.

Under these circumstances, the law may hesitate to make a simple assignment of responsibility. It is not clear what the law will, or should, do when artificial intelligences make mistakes, thereby damaging property, causing monetary losses or killing people. Perhaps we will blame nature or the inchoate forces of the universe. But the legal system is unlikely to rest there; we will not long accept equating the damage done by an unexpected tornado with the mistakes made by programs that are, at some level, human artifacts. If someone *might* have been able to control the outcome of a series of events, the law is likely to be invoked when that control is not exercised. Even with natural disasters, those who could have forecast the path of a storm,³⁰ or warned of the danger of down drafts and wake turbulence,³¹ have been sued. It does not matter that the specific persons responsible cannot be identified. Many legal doctrines assign legal responsibility when it appears that someone, somehow, in some way, was the actual cause of injury.³²

Perhaps we should look to the collection of networked systems and operators in the midst of which intelligent programs do their work, for it is in this ambience that the intelligent program operates. No one system, and no one systems operator, programmer or user, will know the full context of a networked intelligent program—that is precisely why the program was employed, to manage that complexity. Yet where responsibility and thus liability are so spread out over numerous and distributed actors, the very sense of "responsibility" and "liability" is diminished.³³ At that point legal causation seems to fade into the background and tort law, founded on the element of causation, falls away.

Assigning legal liability involves discrimination among an infinite number of causal candidates. That discrimination is avowedly based on perceptions of policy, society's collective sense of what is reasonable and who should be blamed for certain injuries. This article suggests that advances in artificial intelligence, specifically in the distributed computing environment in which such programs will operate, will eviscerate the very idea of cause and effect. Where there are no grounds on which to discriminate between "reasonable" and "unreasonable" candidates for blame, the notion of "legal cause" will fail, and with it the tort system's ability to adjudicate cases involving artificial intelligences.

The legal issues discussed here cannot be solved through new legal tests, criteria or models much beloved of law review articles. The solution sketched out in Part V is more a technological than a legal solution. That is as it should be. Good solutions to problems of advancing technology are those that do not need repeated access to the courts.

II. The Developing Technology

A. Looking Back: The Classic Expert System

At some level, all computer programs make decisions of which the human user is ignorant. These are decisions apparently "made on its own" with some apparent degree of autonomy. The elemental decision tree, if/then statements and branchings in program flow are common to most programming languages.³⁴ These decisions are based the current state of data, which will often be unknown to the user. The temporary state of a variable, data from an input device (e.g., a signal from a remote modem asking for a certain baud rate) and so on, are hidden from the human operator. To those ignorant of the internal workings of the program, it may seem like a "black box," a secret process that magically generates a sensible, context-accurate and apparently intelligent response.

This sort of automated response secured from our day-to-day use of computers now seems ordinary. The program's low-level decision making has in fact simply been built in by the original programmer. Mundane programs, such as word processors, spell checkers, accounting programs and even grammar advisors, are at the lower end of a spectrum of "self-directedness" or automation. These programs produce unexpected results only in trivial ways, such as when our fingers slip as we type. They may be quicker at their limited tasks than humans, and they do not suffer the vice of indolence. But they act only on direct command and use only spoon-fed information.

Next along the spectrum of self-directedness are so-called "expert systems." Various technologies are used to achieve an expert system, some of which can very roughly be termed intelligent. The honorific is used because these systems attempt "to generate heuristics, or rules of thumb, to guide the search for solutions to problems of control, recognition, and object manipulation."³⁵ In short, we term "intelligent" those systems that appear to mimic the higher cognitive abilities of humans.

A wide variety of programming techniques may be applied to the goal of making an artificial intelligence.³⁶ One now-classic technique is a neural network. Trained neural networks contain so-called hidden layers of weighted nodes which interact to generate certain output under various conditions. During a training period of constant feedback from the human "trainers," these neural nets

experiment with various values to their internal nodes, until the net combination of these values generates, in enough cases, the result the trainers desire. The weights on values are then fixed, new inputs are provided and the trainer then expects results comparable to those secured in the training sessions.³⁷ "The problem of learning in neural networks is simply the problem of finding a set of connection strengths [in the nodes] which allow the network to carry out the desired computation."³⁸ The specific configurations of the nodes, and their weights, are irrelevant and usually unknown to the operator.³⁹

For example, one may show a neural net system a series of pictures of faces and tell it which faces are of the same person, or the net may examine a series of pictures of healthy and diseased organs. In each case, the net is told the correct answer during training, and the net then internally adjusts itself to correlate the input with the correct output. Subsequently, the net will take new input, new pictures of faces or of diseased organs, or perhaps numbers from the stock market, and then generate new correlating output as a function of its "learned" internal states. Thus the neural net might conclude that a face was indeed of a certain notorious criminal, that an organ was cancerous or that a stock should be bought or sold. "A neural network can perform highly complex mappings on noisy and/or nonlinear data, thereby inferring subtle relationships between sets of input and output parameters. It can in addition generalize from a limited quantity of training data to overall trends in functional relationships."⁴⁰ Thus, for example, neural nets have acted as expert mortgage insurance underwriters, vehicle operators and sonar processing systems.⁴¹

These neural net systems work as long as the type of input and the general context for the expert system closely parallel the type of input and context of the training sessions. The systems are optimized for specific tasks: for example, playing chess, providing medical diagnoses or adjusting the control surfaces of an airliner. These systems do not function at all in a different context. They do not cause unexpected results except in failure mode. Even though the machine's operators do not know the contents of the "black box," they do know the program's purpose and limits. Obviously, it is wrong to use a car engine diagnostic system to interpret a person's medical condition; it is wrong to use a system expert in the game of Go to make judgments about stock market investments. If loss of life or money result from that "misuse" of a computer system, we know to blame and sue the operator.

These expert systems (whether neural nets or not) are truly machines in the old-fashioned sense. They are housed in specific hunks of metal and silicon, and fed carefully-culled information in meticulously and specifically prepared chunks. These exhibit "intelligence" in only a weak way, regurgitating intelligence, like a child imitating an adult. This lack of meaningful intelligence is patent when programs randomly combine words and then edit those according to rules of programmed grammar to generate "poetry." The same lack of meaningful intelligence is evident in programs which, given the rules of logic, spit out syllogisms, or make a correlation between objects whose common properties were already programmed.

These sorts of programs, predictable, specialized in task, able to use only specially-prepared data, and only weakly intelligent, are not very interesting in the present context.⁴² But, as a result of just those qualities, the programs present no serious conceptual difficulties for the legal system when people lose their lives or property as a result of their decisions. That is, when damage results from the employment of a neural net, it is not difficult to trace back causal vectors to the program, to its trainers/programmers or to its users.

B. Looking Forward: Fluid Systems

Recent work, best exemplified by Douglas Hofstadter and his group at the University of Indiana,⁴³ points to the development of creative, intelligent programs designed to handle radical shifts in context, and thus to produce useful and creative output. They are thus designed to produce unpredictable but pertinent results. These programs, and their expected progeny, deserve to be called "creative" and perhaps "intelligent" in a strong sense.

Hofstadter describes intelligence as emerging from thousands of parallel processes that take place in milliseconds. These processes are generally inaccessible to introspection.⁴⁴ Modeled by computers, such intelligence⁴⁵ is not directly programmed or "hard-wired" (in contrast to many of the "expert" systems referred to above), but emerges as a statistical consequence of the way in which many small program fragments interact with each other. This leads to what Hofstadter terms "epiphenomenal" or emergent intelligence. His programs excel (albeit on a small scale) in *making analogies*; that is, in being able to examine a wide variety of input and relating it in different ways depending on the context, extracting and utilizing different properties of the data from time to time. The context or universe reciprocally derives from the programs' initially tentative, mutable examination of the data.

Hofstadter notes that in the context of human perception, our assumptions are modified by what we see. What we discern is dependent on that context, and on the ability to make analogies to generally persisting analytical structures. For example, as we are increasingly exposed to music, we are able to discern notes and tonalities to which we were previously "tone deaf." Experienced pilots often have unarticulated assumptions about what a cockpit environment should look and feel like, created by sensory input during the course of their flight training. Conversely, a pilot's ability to discern changes in his environment, a musician's ability to discern tonality shifts and a doctor's ability to read an x-ray are all functions of pre-existing knowledge structures. These structures persist until modified by

overpowering new sensory input.

Where humans lack the appropriate pre-existing analytical structure, the data is not recognized as "relevant." In short, perception is a function of making analogies.

The ideal self-organizing network should be able to use context and historical experience to decide what it will learn and ignore. In other words, it should be able to determine for itself what inputs are significant and what are not. In our everyday experience, whether we label a particular piece of sensory information as meaningful or irrelevant often depends on the context.⁴⁶

"[I]n a complex world . . . one never knows in advance what concepts may turn out to be relevant in a given situation."⁴⁷ Intelligent programs must be able to look at data from a variety of perspectives and extract and analyze a variety of properties from the data depending on the nature of the problem to be solved.

The entire system's activity is simultaneously top-down and bottom-up. That is, the input selected as "relevant" from the entire universe of potential data is influenced by pre-existing but not ultimately permanent structures or assumptions, and those structures are simultaneously modified by new input as "learning" takes place.⁴⁸ Contexts used to order new data can be dislodged and replaced under the influence of new input. This becomes increasingly difficult, however, as an existing context gets reinforced through the processing cycle and patterns in data are discerned under the framework of that context. Thus contexts, concepts and perceived data are constantly cycling, slipping past each other and occasionally agglutinating and falling apart, until a "probable best fit," the most unifying answer, is found, or other terminating event occurs. Hofstadter's model of innumerable hidden or unconscious subprograms generating many stochastic micro-decisions without a supervising or top-level executive director corresponds well with other models of mental activity.⁴⁹

But this "probabilistic halo"⁵⁰ model of truly creative intelligence has more important consequences. While the basic statistics do not change, some bizarre and improbable "fringe" responses very occasionally appear on repeated runs of the program.⁵¹ Hofstadter notes that "[i]t is critical that the program (as well as people) be allowed the potential to follow risky (and perhaps crazy) pathways, in order for it to have the flexibility to follow insightful pathways."⁵² The desired strong emergent behavior results when "one or more aspects of [the system's] behavior is theoretically incalculable."⁵³ Specifically, the system may select an inappropriate context and thus may neglect data because the data did not fit a context or analogy that fit other data. Alternatively, the system may reject a good analogy because of peculiar or aberrant data.

It is an essential and, over the long run, entirely assured characteristic of intelligent programs that they will make "pathological" decisions. The nature and timing of these pathological decisions cannot be known in advance. These are not "bugs" in the programs, but are part of their essence. True creativity and autonomy⁵⁴ require that the program truly make its own decisions, outside the bounds expressly contemplated by either the human designers or users. Hofstadter's programs do just that, and I suggest they are the precursors of tomorrow's commercially available AI programs.

C. The Unreliability of Software

The failure of a complex program is not always due to human negligence in the creation or operation of the program, although examples of such negligence are legion.⁵⁵ But, in addition, there are *inherent* problems with software reliability.⁵⁶ While it is at least theoretically possible to *check* to see if a program output is correct in a given instance, it has not been proven that programs can be verified as a general matter; that is, that they are correct over an arbitrary set of inputs. In fact, it appears highly unlikely that even programs which successfully process selected inputs can be shown to be correct generally.⁵⁷

Software reliability generally cannot be conclusively established because

digital systems in general implement discontinuous input-to-input mappings that are intractable by simple mathematical modeling. This . . . is particularly important: continuity assumptions can't be used in validating software, and failures are caused by the occurrence of specific, nonobvious combinations of events, rather than from excessive levels of some identifiable stress factor.⁵⁸

The long-term operation of complex systems entails a fundamental uncertainty, especially in the context of complex environments, including new or unpredictable environments.⁵⁹ That, of course, is precisely the situation in which intelligent agents are forecast to operate.

An excellent overview of the difficulties in checking a program has been provided by Lauren Wiener.⁶⁰ It is, she notes, practically impossible to test software thoroughly. To test a program, all possible sequences of instruction must be run to see if the program in fact behaves as it should. This can take literally thousands of years.⁶¹ For example, assume a stunningly simple program with (i) between one and twenty commands, (ii) that may be called in any order and (iii) that may be repeated any number of times. For a thread (or sequence) of execution one command long, there are of course exactly twenty possible threads. For a thread two commands long, we have 20×20 or 400 possible threads. Those familiar with the mathematics of combinatorial explosions will see this coming. As the number of commands in the thread goes up, the number of threads that need to be run and tested rises exponentially.⁶² Spending one second per test run, it would take 300,000 years to test this simple program of between one and twenty commands. If a bug is found, the entire testing cycle may have to be repeated since a bug fix means, by definition, a program change.

The problem is not limited to what is conventionally thought of as software. The hardware on which these programs run, the processing chips themselves, can be thought of as reified programs, programs encased in silicon. And despite extensive testing by their manufacturers, those chips, too, are buggy.⁶³ It is thus no surprise that, from time to time, software fails, property is damaged and people are killed.⁶⁴ Consequently, programmers just do the best they can. They try to implement rules of thumb and common sense, keeping programs as simple as possible, documenting the code, creating initially stable designs and so on.⁶⁵

We see that it is practically impossible to debug a fixed program with a known range of inputs, on a fixed, unblemished, platform. This article, however, posits the interaction of multiple "intelligent" programs on unknown platforms where none of the programs, operating systems and chip architectures are known in advance where each agent may provide nonpredicted data for input to the other agents which are at that point part of the ad hoc ensemble. It is fair to suggest, then, that although we are assured that intelligent agents will malfunction, we cannot possibly be expected to foresee the nature or timing of the particular problem.

D. Multi-Agent Networked Collaboration

Much has been written on so-called intelligent agents.⁶⁶ These agents are programs originating at one site and executing at a different site. A current model comprises a program written at a local computer, translated into data packets and sent into the telecommunications network, then reassembled at the target computer into a program whose instructions are executed at the target system.⁶⁷ These programs can secure information and then act on it, such as locating flight information and making reservations.⁶⁸

This point is worth re-emphasizing: agents are programs, originating at one site and executing at a different site. Agents are cross-platform compatible; host machines can run agents from other sites regardless of the otherwise incompatible hardware and operating system. Users will often not know when, or where, their agents are executing.⁶⁹

Agents interact with other agents. This is analogous to the way in which mundane programs, such as word processors and telecommunications programs, are comprised of a large number of relatively autonomous subroutines that interact with a variety of other programs, such as macros, spell checkers, file managers and file transfer programs. Agent technology expedites the sharing of large tasks and information, and multi-agent collaboration.⁷⁰ Agents that "know" how to identify and work with other agents-so-called "intelligent" agents-will be more useful and popular than those that do not have such abilities. This ability to collaborate with other agents toward a larger goal may be termed "intelligence."⁷¹ "A group of agents or processes is almost always more successful at solving a problem than single agents or processes working in isolation."⁷²

By design, an agent's creator or user need not know where the agent goes to do its work, or the other systems and agents with which it will interact. This recalls aspects of today's Internet environment, where users do not know which computers are sending them files or which machines are receiving their mail. Users are generally ignorant of (and indifferent to) the multiple programs executing to serve their queries and route their messages.

Current research points directly toward the use of these distributed agents, the *ensemble of which* will achieve the general goals desired by human users. These components will "intelligently" cooperate with each other, securing relevant data from each other, often interacting with each other "in unanticipated ways"⁷³ across platforms and operating systems. "We envision a world filled with millions of knowledge agents, advisers, and assistants. The integration between human knowledge agents and machine agents will be seamless [sic], often making it difficult to know which is which."⁷⁴

Current work at Sun Microsystems on a new programming language known as Java illustrates (and will enable) the movement toward multi-agent collaboration across networks.⁷⁵ Java is derived from the relatively familiar C and C++ programming languages. Java programs spawn small programs or "applets" to be downloaded across the Internet, creating distributed dynamic programmed

content.⁷⁶ Java does this without regard to the platform or operating system of the remote sites; i.e., Java is system independent.⁷⁷ An obvious application is the creation of intelligent agents.⁷⁸ There are other projects for generating distributed objects across the Internet as well,⁷⁹ although with Microsoft's endorsement Java will now probably become the standard.⁸⁰ Users have recently discovered that Java applets may pose serious security problems as they interact with Internet browser software in unanticipated ways.⁸¹

E. Agents and Network Viruses

As discussed above, agents are programs created at one site but designed to operate at other sites. Thus some have noted that an agent is much like a virus.⁸² There is a very thin-some would say imperceptible-line between viruses and presumably "beneficial" programs. "The biggest danger of any network-wide system that allows intelligent agents is that some of the agents will deliberately or accidentally run amuck. Agents have much in common with viruses: Both are little programs that get to seize control of a foreign machine."⁸³

As with "flowers" and "weeds," the definition depends on what one desires. A recent report expressly conflates the two notions:

Now, Japan reports the first "beneficial virus." A report in the *Nikkei Weekly* claims that a group from the Tokyo Institute has developed a virus that travels through computer networks, collects information, spots glitches and reports its findings back to network managers. Wonder if it fetches passwords too.⁸⁴

Many commercial programs have undesirable virus-like side-effects, known by some but not all users: networks can crash, files can be deleted and data can be mutated.⁸⁵

The problem, as always, is that the circumstances of an agent's remote execution, including the other agents with which it may come into contact, cannot be fully predicted by the agent's owner or the remote system's owner. An excellent example was recently provided by the appearance of the so-called "macro" virus. Programs such as the Microsoft Word have a built-in scripting (programming) language that allows the creation of small mini programs known as macros.⁸⁶ These macros simply allow the user to do with one command what might otherwise take a repetitive series of commands. In Word, however, these macros can be hidden inside ordinary data files and then transmitted electronically to other computers where the macro program is executed as the file is read.⁸⁷ An executed macro virus may change screen colors, reproduce itself, add or garble text, or cause other problems. Macro viruses "have the potential to be a much bigger threat than conventional viruses, because people exchange data files all the time With the Internet, the problem becomes much worse."⁸⁸

The point of equating some general programs with viruses is to note that in a complex computing environment, the best-intentioned program may contain modules with unintended consequences. Because the true scope of the relevant computing environment includes an entire network or the Internet (all connected machines), the actions of multiple interacting intelligent agents must be extrapolated in a wide variety of environments.

If our most useful agents-those we send to connected machines to do the most creative work-have undesired consequences operating alone, then the problem will be exacerbated in the context of an express *community* of distributed programs interacting on an ad hoc basis.

We recently constructed a theory of distributed computation This theory predicted that if programs were written to choose among many procedures for accessing resources without global controls, the system would evolve in ways independent of the creator's intent. Thus, even in simple cases where one would expect smooth, optimal behavior, imperfect knowledge in cooperative systems with finite resources could lead to chaotic situations.⁸⁹

Microsoft may have provided us with an interesting precursor of the inadvertent harm that could be caused (or allowed) by an agent-in this case, one that is relatively dimwitted. Users of the Windows 95 operating system found out in the Spring of 1995 that, unbeknownst to them, their operating system contained an agent that was silently reporting back to Microsoft:

Microsoft officials confirm that beta versions of Windows 95 include a small viral routine called Registration Wizard. It interrogates every system on a network gathering intelligence on what software is being run on which machine. It then creates a complete listing of both Microsoft's and competitors' products by machine, which it reports to Microsoft when customers sign up [electronically] for Microsoft's Network Services, due for launch later this year [1995].⁹⁰

Microsoft disputes evil intent, and specifically denied that the Wizard can report on the hardware configuration of every PC hooked up

to a network. Microsoft describes this as just an automated version of a registration card. But the Wizard does appear to detect, and report back to Microsoft, all hardware and software of the local PC.⁹¹ The user of the program is not aware of the scope of the report; rather, the Wizard operates independently of the user for ulterior purposes.

An anonymous Internet commentator reflects:

A friend of mine got hold of the beta test CD of Win95, and set up a packet sniffer between his serial port and the modem. When you try out the free demo time on The Microsoft Network, it transmits your entire directory structure in background.

This means that they have a list of every directory (and, potentially every file) on your machine. It would not be difficult to have something like a File Request from your system to theirs, without you knowing about it. This way they could get a hold of any juicy routines you've written yourself and claim them as their own if you don't have them copyrighted.

The juvenile intelligent agent described here may: (i) perform unauthorized transmission of trade secrets;⁹² (ii) violate federal copyright law;⁹³ (iii) possibly interfere with privacy rights;⁹⁴ and (iv) perform an unauthorized access or interception in violation of the Electronic Communications Privacy Act and related state penal codes.⁹⁵

F. Polymorphism and the Units of Programming Action

The forecast of intelligent agents provided above suggests that agents will constantly mutate to accomplish their purposes, and might even become unrecognizable to their owners. This situation is aggravated by the agents' viral aspects noted above. Some viruses, both biological and digital, mutate to survive by defeating immunological and other weapons designed to destroy the viruses; it is obviously more difficult to defeat an enemy that constantly changes its guise. The assumption that intelligent programs will mutate to accomplish their purpose is strengthened by references, conscious or not, to recent work in the artificial life context. For example, interesting recent developments suggest the self-modification of code and the creation of program elements which, left to their own devices and evolution, may result in exceedingly efficient programs that humans alone could never have created.⁹⁶

The intelligent agents forecast above need not be able to change or adapt in quite that manner to accomplish their purposes. However, this discussion does suggest a larger problem of mutation or "polymorphism" in programs. Polymorphism can be confusing because in the digital context it is an ambiguous notion. Specific bits of code (we might call them codelets, as Hofstadter does) may or may not change as a larger chunk of program mutates. The larger the program, the more likely it is that as processing takes place "the program" can be said to change in some way. Many "programs" are in fact composed of dozens of discrete codelets and data sources such as dynamic link libraries (DLLs),⁹⁷ initialization files and device drivers,⁹⁸ with logical program flow depending on all of these subsidiary structures. These structures, in turn, can be shared among a number of encompassing "programs."⁹⁹

While we use the term "program" for convenience, it is more accurate to think of a "processing environment" containing both data and instructions (or a "logical program flow" depending upon both data and instructions), because no fundamental distinction can be drawn between the two. Computers see data and instructions in the same binary way. Data can act as instructions and vice versa. A program can accomplish the same result by modifying data and feeding them to a static program, or by modifying a program and feeding it static data.

Today, object orientated programming (OOP) is common; for example, Borland's and Microsoft's C++ programming language provides OOP. This language makes express the community of data and instructions. A so-called object is a bundling of "data structure and the methods for controlling the object's data."¹⁰⁰ These objects will, at least in C++, always contain both the data and the data's behavior rules. These two are "encapsulated" in the object. As an elemental programming unit, this object can then be duplicated, modified and used in an infinite number of contexts. These objects, which are types of programs or codelets, may or may not change independently of the larger program.¹⁰¹ Often the properties of the object—that is, its combined data and instructions—are invisible to the programmer and even more so to the program's ultimate user.¹⁰²

A putatively static program may operate in a dynamic (data) environment. In such a case, referring to the processing environment at large, we could as correctly classify the program as dynamic or polymorphic. When the context of a constantly mutating data environment is included in the conception of the processing environment, it becomes apparent that *all* programs should be thought of as polymorphic.

There is, to be sure, a forceful convenience in naming programs and assuming their identity and persistence through time. Without

such assumptions of persistence and identity, copyright and other legal doctrines affecting software would be difficult to conceive. Most of those doctrines, after all, are based on traditional property-based notions, and they depend for their sense on persisting, identifiable, and preferably tangible, property.¹⁰³ But those assumptions are just convenient fictions. As with other legal fictions of persisting entities (such as corporations), the fictions succeed to some extent, and for some purposes.

However, to the extent that the criterion of continuity and the true behavior of software agents diverge, the utility of the fiction of persistence dissolves. Even today many problems resulting in data loss and interruptions in computer services are clearly caused by ever-shifting interactions among the subsidiary "programs" or data files referred to above, such as initialization files, DLLs and so on, which operate independently.¹⁰⁴

In brief, processing environments are polymorphic and distributed. The environments change over time and overlap in time and in space (computer memory). We will see¹⁰⁵ that these characteristics pose great difficulties for a traditional legal analysis of the effects of such processing environments.

G. Summary Forecast

Before we move to the legal domain, we should summarize the forecasts.

We should assume that:

- (i) we will employ intelligent agents specifically for their creative intelligence and corresponding judgment;
- (ii) these agents will make decisions,
- (iii) as the consequence of the interactions of a distributed ensemble of such agents,
- (iv) which ensemble will be comprised of polymorphic programs; and
- (v) some of the resultant decisions will be "pathological"; that is, not only unpredictable and surprising, which may be desirable, but also having unintended and destructive consequences both for networks and for the substantial portions of our infrastructure connected to those networks.

The question of liability for these unintended but inevitable injuries is far from simple, and cannot be solved under current legal doctrine. As discussed in the next section, "causation" analysis of these injuries is particularly difficult.

III. Causation in the Legal Domain

Men are not angered by mere misfortune but by misfortune conceived as injury.¹⁰⁶

Where injury is done, we suppose the law should intervene. The legal system fixes liability on those responsible for the injury, the so-called "legal cause" of the injury. The drunk driver is the cause of the car accident and should pay the victim he hits. Perhaps the aircraft manufacturer is liable if the aircraft is negligently designed and crashes. But deciding under what circumstances to hold people, corporations and others liable can be difficult.

Other elements are, of course necessary; no one wins a case just because he has sued the "legal cause" of his injury.¹⁰⁷ But causation is a necessary element of any civil tort lawsuit, and without that factor the plaintiff's case falls apart. Thus this article evaluates the central issues of causation.

A. The Conundrum of Causation

The problems with causation can be illustrated in a series of examples. Do we hold the bartender liable for the drunk driver's injuries (to himself or others) when everyone knew he drank twenty beers and then loudly announced he was driving home?¹⁰⁸ When an aircraft crashes, we may hold liable the engine and airframe manufacturers, but how about the air traffic controller who saw the rainstorm on the screen and never issued a warning? When a car thief knocks someone down with his stolen car, do we hold the owner liable because he left the car unlocked and the keys in the ignition?¹⁰⁹ Do we hold tobacco companies liable for the cancer death of a smoker? Should we hold an armed bank robber liable for murder if an accomplice pulls the trigger in the heat of a robbery gone

wrong? Should we hold a robber liable when a *police officer* shoots someone?¹¹⁰

More difficult are issues presented by a focus on alternative and concurrent causation. For example, two people fire the same type of weapon in the direction of the victim, and one bullet—we don't know whose—hits the victim and is fatal. Is either shooter liable? Both?¹¹¹ How about when a drowning results from the victim's inability to swim, someone's failure to properly supervise the children, another's failure to call for help, and someone else's kicking the victim into the water?¹¹² Ultimately, cases which appear to deal only with how far back to go in a chain of causation in actuality often present facts which suggest multiple concurrent (or nearly concurrent) causation.

From the difficult issue of concurrent causation, we turn to a more complex subset, where it is not clear when a force, person or influence was the "cause" of injury. Who, if anyone, is liable when decades after the fact it turns out that a certain type of insulation kills people who used it?¹¹³ Should *anyone* be liable when a farm pesticide, decades after it was applied to fields, is found to cause a statistical increase in the odds of getting cancer for those living in the area? Should the chemical company be liable? The farmers who applied it? The government agency that approved it? Suppose the only entity that knew the pesticide was dangerous happened to be a trucking company that shipped it—do they pay for the damage? Is the Coppertone suntan cream company responsible for millions of skin cancer cases because, for years, their advertisements encouraged people to tan their skin as dark as possible?

In this last set of difficult questions the harm is a function of many factors. We may not be able to determine if the harm was caused by human or natural agencies; rather, the congruence of human, natural and technical agencies caused the ultimate harm. Which one do we pick out as the *legally responsible* cause? Or should we blame all the causal vectors, and make every human and corporate actor responsible for the unanticipated injuries? We could, of course, blame no one, call the event an "Act of God," and rest the causal "responsibility" with natural forces.

In the abstract, the law provides no good answers to these questions, because no *general* rule exists on how to pick, out of the infinite mass of all possible factors, the one (or two) on which legal blame is fixed. To illustrate, imagine that a factory burns to the ground when a match is lit and ignites the surroundings. Obviously the match, and the person who lit it, are the cause of the destruction. But assume this is a match-testing factory, in which for years billions of matches have been tested in a fire-proof room filled with inert gases. A mistake is made, oxygen leaks in and the factory explodes. We have the same causes at play, but we are tempted to identify as the "cause" not the match, but rather the gas leak.¹¹⁴

B. The Tort System's Causation Analysis

1. Cause In Fact

The traditional tort system first looks for "causation in fact" to resolve the conundrums posed by the examples above. Some chain of circumstances that connects the accused's acts and the injury suffered may constitute a cause in fact, no matter how tenuous and no matter how many other contributing or intercepting forces there may have been.

That factual inquiry is essential, and it is worth noting that even theories of strict liability do not tamper with that essential inquiry. Strict liability, to be sure, dispenses with most classic elements of a tort, such as fault, negligence or recklessness, or other measures of an accused's culpability.¹¹⁵ But causation in fact is always required. This requirement, roughly approximated by establishing that the accused's acts were at least the "but for" cause of the injury, is a fundamental, intractable element of proof of a tort.¹¹⁶

There are, to be sure, certain epoch-making cases which appear to undercut this requirement, but in fact they do not. For example, in *Summers*¹¹⁷ only one of two shooters fired the wounding bullet; the court decided that unless one of the shooters could prove his innocence, both shooters would be liable. Plaintiffs ever since have tried to expand the rationale to other contexts in which they have difficulty pinpointing the actual cause.¹¹⁸ True, a *Summers*-like doctrine may have the effect of holding liable one who actually was not the "cause in fact" of the injury. But it is important to note that the case simply switched the burden of proof on causation from the plaintiff, who is generally a person in a good position to know what happened to him, to the defendants who *in that case* were thought to be in a better position.¹¹⁹ Causation in fact was still an element in *Summers*. The only difference from a typical tort case is that the defendants were saddled with the burden of proving its absence. As other cases have since noted, that switch of burdens of proof will not often be permitted.¹²⁰

And then there are cases such as *Sindell v. Abbott Laboratories*.¹²¹ *Sindell*, too, formally only shifted the burden of proof on causation. The court held that where some type of fungible good (the drug DES in this case) made by a plurality of potential defendants *in fact* caused injury to the plaintiff, then the odds that a given defendant's products caused the injury were equal to the percentage of the

market for the fungible good held by that defendant.¹²² That "market share" is then used to calculate the percentage of the damages for which that defendant will be liable.¹²³ Crucially, *Sindell* acts only to shift the burden. Defendants still win when they can show that their products, as a matter of fact, could not have been responsible for the injury.¹²⁴

2. Proximate Cause .;

But tracing back through a chain of causes is not enough for liability. Indeed, the confounding examples provided above *all* assumed that some chain of events in fact links the potential "cause" (such as a match lighting or the structural condition of an aircraft wing) with the harm (the exploding factory or crashed airplane). Quite aside from this causation "in fact," a wholly separate issue remains: legal or proximate cause. Proximate cause is the vehicle used by the law, in its wisdom and omniscience, to carry out society's policy by holding a certain agent liable. A court may hold some agents liable while excusing others who are also *in fact* linked to the disaster.¹²⁵ The doctrine of proximate cause is used to select, from all the causes in fact, the entity(s) which will be held responsible for the injury.

This issue of *legal* responsibility, or *legal* causation, depends on the context. That is, do we look at just the match and its user? Or do we evaluate the larger context of the gases and atmosphere, and so perhaps sue the provider of the inert gases? Do we blame the materials of which the factory was built, with the possibility that the architects and builder are liable? Do we look at the entire idea of having a match-testing facility, and suggest that it is inherently so dangerous that *anyone* who operates such a plant will be liable for any damages, without regard to negligence, fault or anything else? Do we sue the governmental entities that knew about the plant and did nothing to stop it?

The decision on which of the many contexts to use to evaluate liability depends on the sort of policy to be furthered by the liability rule. We presume that socially desirable conduct is furthered by singling out certain actors from the rest of the context, from the rest of all the other possible causes, and then punishing them as transgressors. As the canonical text on the subject states, a legal or "substantial" cause is that which "reasonable men" would agree is a cause, "using that word in the popular sense."¹²⁶ Holding an entity liable is a statement, by the law and the culture it protects and in part defines, that those entities can and should bear responsibility for avoiding the harm at issue.¹²⁷

Courts formally accommodate shifting public policy, changing mores and technical developments by asking whether an injury was "reasonably foreseeable."¹²⁸ Reasonable foreseeability is essential for establishing proximate cause. One who reasonably should have foreseen a consequence, and was in a position to prevent it, may be liable for it. If an earthquake is reasonably foreseeable, then the architect may be liable for the house flattened by the tremor; if a flood is reasonably foreseeable, then the law imposes liability on the builders of the dam which fails.¹²⁹

A fitting example of proximate cause's reliance on foreseeability is provided by the doctrine of *intervening* and *superseding* cause.

An intervening cause, as the term suggests, is simply one which intervenes between the defendant's act and the injury. When a contractor builds a weak foundation and an earthquake shakes it, loosening a brick and killing a nearby pedestrian, the earthquake is an intervening cause between the negligence and the injury. Courts decide if the intervening act was sufficiently *unforeseeable* so as to constitute a *superseding* cause-itself a wholly conclusory term-which would free the defendant of liability. Intervening criminal conduct may or may not be "superseding"; it depends on whether the court thinks the intervening criminal act ought to have been foreseen by the defendant.¹³⁰ Intervening "natural" forces such as floods,¹³¹ power outages,¹³² excessive rain,¹³³ wind,¹³⁴ and fog¹³⁵ may or may not be treated as "acts of God" for which no human agency is liable. Liability will not depend on whether humans were involved in some capacity (for they always are, in all these cases), but only on whether the court believes that the action of the natural force should have been foreseeable.¹³⁶

Because foreseeability is a matter of policy and perception, these things change over time. Bars once were not liable for drunks they watched drive away; now they may be. Landlords once were not liable for rapes on their premises; now they are in some cases.¹³⁷ Different views espoused by different judges on what should have been foreseeable have led not only to defendants' exculpation in favor of an "Act of God," but also to liability for natural catastrophes such as erosion, where defendants did nothing to cause the injury and there is not a whisper of evidence that they could have done anything to prevent it.¹³⁸

What is "reasonably foreseeable," and so what qualifies as a "proximate cause," depends on custom and what people generally believe. These in turn may depend on general impressions of what technology can do. For example, it is "reasonably foreseeable" that shareware downloaded from the Internet may be infected with a virus-foreseeable to some, but not all. It is foreseeable that sensitive information transmitted via cellular phone an unsecured radio link can become public-foreseeable to some, perhaps, but not to a

President of the United States and certain British royalty.¹³⁹ In each case, the courts do not focus on what individuals—even the individuals involved in the case—knew or expected, but rather on what should have been expected by a "reasonable" person, an abstract person compiled by a court from the whole population.¹⁴⁰

What is deemed "reasonably foreseeable" depends on the court. Reading the tea leaves of contemporary society, different judges have different opinions. What is not reasonably foreseeable to one judge is perfectly predictable to another. A superseding event in one state may not be in another.¹⁴¹ Reasonable foreseeability is a moving target; it dodges and weaves depending on public policy, and on the perceived technological sophistication of the population. What is reasonably foreseeable depends on the sense of juries and judges, presumably reflecting what their culture believes is "reasonable." Thus the test acts as a mechanism by which the judicial system, in theory, remains responsive to its social environment. In the context of intelligent agents, the actions of the program or its progeny raise serious questions about the "reasonable foreseeability," and thus causation, of the harms these programs will do.

IV. Causation In The Digital Domain

Once a computer system is designed and in place, it tends to be treated as an independent entity.¹⁴²

Diffusion and abandonment of responsibility

[S]upervisors may eventually feel they are no longer responsible for what happens—the computers are.¹⁴³

Treating computers as responsible agents may mask the human authors of the mischief which may result from the use of computer systems.¹⁴⁴ Winograd and Flores provide the example of a medical diagnosis machine, which may be "blamed" for an erroneous diagnosis through plain *human* error: i.e., the system is used in the wrong context, or programmed with assumptions not shared by the users, and so on. The authors suggest that those human users and programmers are responsible, not the machine.¹⁴⁵ However, in the complex processing environment envisioned in this article, one cannot impose liability on any identifiable agency, human or otherwise.

This is the case because multiple agent systems imply at least the causal input of multiple independent programmers of the basic scripting or authoring software, a vast number of users creating distinct intelligent agents, and an unpredictable number of agent to agent interactions on an unpredictable number of interwoven platforms, operating systems, distributed data and communications programs, each of which in turn incorporates at least some further limited programming. This inevitable causal complexity poses problems for traditional tort law, in which a determination of proximate cause is essential, as it evaluates the liability of an intelligent machine system.

A. An Example of An Intelligent Processing Environment

As we illustrate an intelligent system, we recall the comments of researchers quoted above: "We envision a world filled with millions of knowledge agents, advisers and assistants. The integration between human knowledge agents and machine agents will be seamless [sic], often making it difficult to know which is which."¹⁴⁶ This article assumes that intelligent agents will be created both by humans and by other intelligent agents, as needed.¹⁴⁷ Humans can now use programs such as Telescript and Java¹⁴⁸ to make intelligent applets.¹⁴⁹ These agents, created at diverse computers, will communicate with each other, to accomplish larger goals. Even now small programs (fondly referred to as daemons) stand ready to assist those involved in telecommunications sessions. They can inform the calling system that a connection is available, receive a request from a remote user, and service the request by shifting directories, sending data and so on.¹⁵⁰ Other programs such as Java know how to manipulate these daemons, and we can expect that all these programs will know how to interact with other programs such as Internet search engines,¹⁵¹ graphical web browsers such as Netscape, intelligent natural language query tools, and a host of other useful programs resident in one machine or another. No one of these small programs could accomplish the ultimate goals of a human user, but together their actions may *in the aggregate* appear intelligent in a broad sense.

1. The Structure of Alef

To illustrate some attributes of the processing system in which intelligent agents will work, imagine a hypothetical intelligent programming environment which handles air traffic control, called "Alef."¹⁵² The description that follows emphasizes the networked distribution of agents, their unpredictable variety and complexity, and the polymorphic ambiance of the intelligent environment as a

Alef's fundamental task is to guide the flow of air traffic in and out of a section of space, including airports in the area. Today, this work is in fact accomplished by thousands of intelligent agents (both human and computer) in airplane cockpits, airport control towers, air traffic control centers and weather forecasters. There are a number of modules, or agents, that would be a part of any large-scale intelligent processing environment. Alef is an example of a large scale processing environment composed of many modules. Those modules within Alef are given here in broad, sometimes overlapping strokes, with examples specific to Alef's tasks. Many of these modules would in fact be composed of discrete sub-modules.¹⁵⁴ Examples of specific tasks within Alef which would be performed by such sub-modules include:

Sensory input: Alef will use voice, visual and motion information, as well as traditional data input. Voice and motion recognition software, as well as software to accurately read and analyze digitized visual input, are complex and sophisticated. Each contains subsidiary modules. Alef would secure information from airport surface sensors, radar and human voices.

Sensory control: Alef may need to turn on and off lights, open and close microphones, modify the sensitivity of and/or select various sensors and undertake robotic actions.¹⁵⁵

Data input: Alef will need to utilize text files, knowledge bases¹⁵⁶ and other databases. Alef should know the "rules of the road" such as the federal aviation regulations, facts about the aircraft it is guiding (such as speed, position, fuel, load, range, altitude, destination and origination, as well as aircraft capabilities such as maximum rate of climb and engine-out glide distances), weather information and so on. The data reside in hundreds of physically scattered sites.

Heuristic analysis: Alef should have access to expert systems, rules of thumb and heuristics. These would, for example, recommend separation between various types of aircraft (over and above the minimum requirements of FAA regulations), and which approaches, departures and runways to use depending on weather, general traffic and facts specific to a given aircraft (such as its destination).

Basic assumptions: Alef should have assumptions about the way the world works: a grasp of gravity and of the fact that two airplanes cannot inhabit the same space are the most obvious.

Goals and commitments: Alef may be equipped with certain goals, such as minimizing the rate of fuel burn and the amount of flight time en route. Alef should possess a preference for rapid responses to certain types of urgent communications from aircraft. Also, Alef may be committed to neighborhood noise abatement involving difficult or circuitous approaches. Or it may prefer easy or simple instrument approaches to those involving many way points, complicated turns or holding patterns.

Basic computing, load-sharing and load distribution: Alef must have the capability to distribute its processing requirements, and then integrate the results. This is not a trivial task. Alef must constantly manage data that are processed by a variety of distributed programs, in the context of rapidly changing demands for processor and memory resources in different locations.

Resource management: Alef must have power management to ensure continuous operations in emergency situations and guaranteed access to the communications networks, such as telephone, cable, satellite and microwave channels, to ensure required communications among agents. Because it is essential to keep Alef operational, Alef may have agents devoted to preemptive or dedicated access to power and communications facilities.

Programming: Alef may find it expedient to make new subroutines. This would have the effect of modifying the substance, or the actions, of other agents. Obviously, agents useful only under certain circumstances (e.g., in low visibility conditions) could be turned off altogether when those conditions did not obtain; other agents might be modified, such as those controlling power requirements, those that deal with information overload or those that adjust the intensity of runway approach lighting in reaction to weather conditions. At a lower computational level, Alef would also manage its memory and other systems tasks.

Communications:¹⁵⁷ Alef should be able to accomplish routing of messages (data), registration (address identification) of the other agents in the environment, and mediation and translation of messages among the various agents. Alef would also communicate with other intelligent systems controlling aircraft within their jurisdictions across the United States and abroad.

Interface:¹⁵⁸ The intelligent environment will interact with humans. Some of these agents may also be categorized as sensory input agents. Alef will use graphical user interfaces and other programs to communicate with humans on the ground and in the air. Thus, Alef will often communicate with human pilots, although in emergency situations Alef may directly control the flight

of an aircraft itself.

2. *The Operation of Alef*

a. Routine Operations

Alef encompasses the plurality of agent types outlined above. Each agent is connected to a variety of similar agents, as well as to many wholly distinct agents. Agents run in-and transmit themselves between-a variety of locations. In the Alef example, agents may be found on board hundreds or thousands of aircraft, on vehicles and other sites on airport surfaces, in satellites, at control towers and in more centralized computing systems. Agents are continuously added in and taken out of the mix. The sources or architects of these agents vary widely. Some agents will have been written by aircraft and radar manufacturers, some by pilots programming their own aircraft,¹⁵⁹ some by certain of Alef's own agents and others by human operators in control towers, at air traffic control centers and at weather observation facilities. The data streams into Alef will be truly staggering, including changeable weather and highly variable data on each aircraft in Alef's space or about to enter its space. Managing this complexity of data is precisely the rationale for Alef's existence.

Alef routes air traffic. In emergencies, as Alef perceives them, Alef may control individual aircraft to prevent collisions. That, and Alef's more mundane actions, would be a function of the totality of the agents both then currently active and previously active which modified data or which modified other linked agents.

To what appears to a human to be repetitions of the same problem, Alef may arrive at different responses. For example, when guiding an arriving jet, Alef may route it over one way point at one altitude one day, and by a different route the next day. Communications processing and other tasks would differ day to day, and from moment to moment. Mandating one approach or the other would defeat Alef's purpose.

b. Pathological Operations

Alef's pathological operation cannot specifically be forecast. But if Hofstadter¹⁶⁰ is right, some pathology will erupt as a function of Alef's attempt to solve a new problem by analogy to an old one. The analogizing process will require Alef to combine its constituent distributed agents in new and interesting ways.¹⁶¹ The result will be an unexpected configuration or interaction of agents, including perhaps new agents created to handle the new problem. An anomalous situation may cause an anomalous result; based on its prior experience, Alef may decide to ignore certain data and attend to new information. Alef may use the "wrong" analogy to incorporate data. For example, a rapid decrease in an aircraft's altitude may be seen as evidence of fuel problems, and not the near-miss air collision it was. In short, Alef may be seen in retrospect as having exercised poor judgment.

Specifically, Alef may ignore certain warnings. For example, noting that ground or collision proximity warnings activate long before a collision is actually imminent, Alef may route traffic closer to obstructions than it should. Perhaps in its enthusiasm to ensure continuous power, Alef reroutes electrical power away from the surrounding homes. In an effort to streamline operations, Alef may encode data under its control, effectively making it unreadable to humans and useless for other purposes. Perhaps the system misunderstands communications from a pilot or from instruments on board an aircraft, or assumes the pilots have information they do not, and as a result a plane runs out of fuel. The specific admixture of agents that generates the poor judgment is more likely to occur at a lower, and less overt, computational level. Conflicting claims upon its computing or memory management resources may cause Alef temporarily to drop communications with an aircraft at a moment that turns out to have been critical.

Of course, each of these problems can be fixed. Alef can simply be overridden with instructions never to deviate from pre-assigned clearances between aircraft, never to modify its access to electrical power, never to tamper with the format in which its data is held, and so on. Global constraints can easily be imposed, once we know of the problem. These will prevent systems from "evolv[ing] in ways independent of the creator's intent."¹⁶²

But we do not know the problems in advance. Experience is the great teacher, and sometimes the experience will come first to the intelligent system, *before* humans can accommodate the crisis. Global constraints defeat the purpose of intelligent systems such as Alef. To be sure, these constraints are a matter of degree.¹⁶³ However, the more global constraints control the decision-making ability of the system, the less the system's intelligence is a function of its ensemble of collaborative agents. The environments discussed in this article do *not* have a "central reasoner."¹⁶⁴ For a Hofstadterian program, the guiding analogy is not presented in advance; the intelligent system detects or invents it. Nor do these systems exist aside from a larger networked environment. Thus, it is meaningless to suggest, perhaps as another form of global control, that the system be severed from a network and neatly encapsulated into a stand-alone box. The imposition of such global constraints would not simply modify the behavior of these intelligent systems, but would

eviscerate them.

B. Unpredictable Pathology: Absolving Humans of Liability

If and when Alef, as an entire processing system, "causes" an unanticipated fault, it will be difficult to establish human liability. For it is not enough that some undifferentiated damage of *some* kind may be expected. We "expect" those problems, surely, from many software environments.¹⁶⁵ Rather, liability attaches to those who reasonably should have foreseen the *type of harm* that in fact results. That is how the "reasonable foreseeability" test is implemented. Liability does not extend to the "remarkable" or "preposterous" or "highly unlikely" consequence.¹⁶⁶ Of course, this will always be a matter of degree. Consequences may not be *so* unforeseeable as to relieve the causative agents of responsibility.¹⁶⁷ As noted above, the bounds of foreseeability fluctuate, varying from court to court, and over time.

The inability to pinpoint specific human responsibility for failure suggests that "the machine" or the network "system" should be blamed for damage it causes. The temptation to treat sophisticated intelligent agents as independent legal entities, thus absolving the humans involved, is powerful. The agents appear to be autonomous and "independent"; their pathological results are by definition unpredictable. No human will have done anything that *specifically* caused harm, and thus no one should be liable for it. Just as we are not liable for the consequences of a human agent's unforeseeable pathological actions,¹⁶⁸ so too humans should be absolved of liability for the unforeseen results of machine intelligence's pathology.

Humans are, of course, held legally responsible for some computer malfunctions. As noted in the discussions of classic expert systems, many "computer errors" are easily traced to human negligence or other fault. It makes good sense to impose liability on those persons who have the power to avoid the injury in the future.¹⁶⁹ But that rationale for imposing liability fails when no particular human has the ability to prevent the injury, short of banning the use of intelligent agents altogether.

Liability in the computer context must depend, as it does in other contexts, on plaintiffs' ability to convincingly argue that a given injury was "reasonably foreseeable." This means foreseeing the context of an action (or inaction), and being able to predict at least in general terms the consequences of the action on the context.¹⁷⁰ But even today it is often difficult to charge people with predictive knowledge of the electronic context and of the consequences of various deeds on that context.¹⁷¹ For example, a hard drive's (programmed) controller works well in most circumstances, and generally no data is lost. It turns out that with the advent of new 32-bit operating systems such as IBM's OS/2 Warp, an attempt to process multiple interrupts causes the controller to occasionally corrupt data stored on the drive with possibly catastrophic consequences. Is this a case of a "flawed" controller, as the media suggests? Or is this a flawed operating system? After all, not all 32-bit operating systems cause this data corruption.¹⁷² Perhaps both the maker of the drive controller and of the operating system are eligible for "blame" here. Yet no "reasonable" person would have predicted the cause of the failure, so there is no basis on which to hold a given maker liable. The problem is simply unforeseeable incompatibility in the linked systems.

This problem becomes more acute with artificial intelligences such as Alef which operate in environments in which elements are continuously added and dropped out. The specific pathological judgment calls made by Alef are not "reasonably foreseeable," and thus courts should treat them as superseding causes, corresponding to unexpected fog or storms which, in former "natural" contexts, eluded human responsibility.

C. Failure of Causation: Absolving Programs of Liability

We are beginning to work in a class of fundamentally non-linear media . . . [and] the concept of logic pretty much goes out the window. What's going to replace it?¹⁷³

Fixing liability on networked systems will solve no problems. Social policies are not furthered when courts decide to blame an ensemble of networked agents. The law does not yet recognize programs as legal entities capable of defending themselves or paying damages.¹⁷⁴ Targeting intelligent agents for legal liability fails to solve two fundamental and interrelated problems. First, interacting data and programs "responsible" for a failure are as distributed as the involved human agencies. Second, classic cause and effect analysis breaks down.

The dispersion of the agencies involved-human and otherwise-has been described above.¹⁷⁵ These agencies do not simply trigger each other sequentially, in a specific time or space order. We may have multiple concurrent causes, but there is no mechanism for selecting out those which are legally "substantial" (and hence lead to liability) from those that are in some fashion incidental. Over time, the configuration of active elements shifts: different agents act on mutating data, and different sets of agents interact from moment to

moment. In an eternally changing context, agents have no inherent substantiality or persistence. They are polymorphic. The agents' roles change from centrally active, to sustaining context, to inactive or absent altogether from the processing environment.

The notion of "proximate" or "legal" causation implies a court's ability to select out on a case-by-case basis the "responsible" causes. But where damage is done by an ensemble of concurrently active polymorphic intelligent agents, there is insufficient persistence of individual identifiable agencies to allow this form of discrimination. In this context, there will generally be no room for the courts to use social policy to distinguish among the programs and operating systems, encapsulated data, objects and other data-cum-program entities each of which is a cause in fact (and perhaps a cause *sine qua non*) of the injury or damage.

On what should society and the courts focus? A court may have a felt sense that it should hold the manufacturers of exploding vehicles responsible for resultant injuries, even when there are a series of other eligible causes (cars are driven too fast, tank manufacturer made the gas tanks out of thin metal, etc.). Likewise, a court may hold accountants and lawyers responsible for allowing a company to issue a false securities prospectus because they should have foreseen the injury. A court makes choices in an effort to fix blame where it feels a need to compensate the injured and to encourage desirable behavior by those able to foresee the harm they may cause. But where "fault" is a function of many ephemeral agents' unpredictable interactions, there is no target for the judgment of social policy.

What I have termed pathological outcomes are *inherent* in the electronic cosmos we are making. Pathology is both "immanent and elusive from an excess of fluidity and luminosity."¹⁷⁶ "[V]ery strange answers"¹⁷⁷ are exactly the sort of responses we expect from our agents, and it is unlikely that we will ever arrive at a stable global electronic ecology that buffers us from these strange answers. Indeed, it is likely that networks will become increasingly linked, reactive and complex, and so therefore increasingly entrusted to the day-to-day management by agents. In this sense, error is emergent, caused by all and none of the transitory participating elements.

No surgery can separate these inextricably entwined causes. No judge can isolate the "legal" causes of injury from the pervasive electronic hum in which they operate, nor separate causes from the digital universe which gives them their mutable shape and shifting sense. The result is a snarled tangle of cause and effect as impossible to sequester as the winds of the air, or the currents of the ocean. The law may realize that networks of intelligent agents are not mysterious black boxes, but rather are purposeful, artificial constructs. But that will not solve the problem of legal liability. The central doctrine of proximate cause, essential in the sorting out of multiple causes and tagging some in accordance with public policy, is useless when causes cannot be sorted out in the first place.

V. Turing, Where Angels Fear To Tread

The electronic universe is multiplying at an exponential rate, absorbing many segments of the infrastructure, and at least indirectly controlling much of the rest. In that utterly dispersed place, traditional doctrines of legal liability, such as cause and effect and foreseeable injury, do not exist. This will shock not only lawyers and judges, but also the businesses that expect to be compensated for their efforts and the injuries they suffer. If we try to use a system based on sorting through various causes to navigate a miasma where "cause" means so little, the legal system and the rapidly developing technology will suffer. People will be unfairly charged for damages they could not have prevented or previewed. Extravagant laws passed by legislators unclear on the technology will scare developers, systems operators and resource providers from fully engaging in the commerce of the electronic universe. If there were an alternative that plainly recognized the impotence of the traditional tort system, we might avoid some of these frantic and destructive efforts to control the uncontrollable. To that end, I propose the Turing Registry.¹⁷⁸

A. The Registry

The behavior of intelligent agents is stochastic. Like risks underwritten by insurance agencies, the risk associated with the use of an intelligent agent can be predicted. This suggests insuring the risk posed by the use of intelligent agents. Just as insurance companies examine and certify candidates for life insurance, automobile insurance and the like, so too developers seeking coverage for an agent could submit it to a certification procedure, and if successful would be quoted a rate depending on the probable risks posed by the agent. That risk would be assessed along a spectrum of automation: the higher the intelligence, the higher the risk, and thus the higher the premium and vice versa. If third parties declined to deal with uncertified programs, the system would become self-fulfilling and self-policing. Sites should be sufficiently concerned to wish to deal only with certified agents. Programmers (or others with an interest in using, licensing or selling the agent) would in effect be required to secure a Turing certification, pay the premium and thereby secure protection for sites at which their agents are employed.

An example may help here. Assume a developer creates an agent that learns an owner's travel habits and can interact with networked agents run by airlines, hotels and so on. The agent is submitted to the Turing Registry, which evaluates the agent's risk. How likely is it to interact with agents outside the travel area? Does it contain a virus? How responsive is it to remote instructions? How quickly does it cancel itself when launched into another network? Does the agent require supervisory control, and to what extent? What decisions

can it make on its own? Generally how reactive is it? Agents designed to interact or capable of interacting with nationwide power grids or nuclear power plant controllers would be assessed a correspondingly high premium. In all cases, the premium would be a function of the apparent reactivity of the program, its creativity and its ability to handle multiple contexts-in short, its intelligence. High intelligence correlates with high risk. Such intelligence and corresponding risk is associated with agents designed to interact with a wide variety of other agents and which, concomitantly, are not designed to submit to global controls.¹⁷⁹ The Turing Registry predicts the risk posed by the agent, and offers certification on payment of a premium by the developer.

The Registry will certify the agent by inserting a unique encrypted¹⁸⁰ warranty in the agent.¹⁸¹ United Airlines, Hilton Hotels, travel agents and so on would express trust in the Registry by allowing into their systems only Registry certified agents. The encrypted warranty may also be used to ensure that certified agents only interact with other certified agents.

At some point a pathological event will occur. Perhaps United's data base is scrambled. Perhaps an intelligent agent, trying to do its best, finds out how to re-route aircraft to suit a trip request, or it blocks out five floors of a hotel during the Christmas season-just to play it safe.¹⁸² At that point the Turing Registry pays compensation if its agents were involved, without regard to fault or any specific causal pattern. That is, if the Turing Registry determines that a certified agent was involved in the disaster as a matter of cause *in fact*, the Registry would pay out. No examination of *proximate cause* would be undertaken. The Registry would provide compensation if and when Alef tampered with data in linked systems. And without an impossible investigation into the specific agents directly responsible, the Registry would pay out if an aircraft under Alef's control was diverted into a mountain. The Registry might pay for mid-air collisions, the consequences of data and program modules residing in computers owned by the Government at air traffic control centers, by private pilots and by companies providing various navigation and data management services.

Risk might be lowered with minimum requirements. Because agents' environments are just as important as the agent itself,¹⁸³ perhaps the Registry may not pay compensation unless at the damaged site (i) certain environmental controls are in place, designed to curb agents and limit their reactivity and (ii) essential security, data backup, and other redundancy systems. Perhaps the Registry would insist that covered sites *only* allow in registered agents. But the imposition of substantial global controls will defeat the purpose of the intelligent system, and thus such requirements offer very limited protection.¹⁸⁴

Insurance companies already exist, as do companies that test programs for compatibility with a variety of hardware and software. Some recently formed companies specializing in electronic security will probably escrow the electronic "keys" needed to access encrypted data.¹⁸⁵ Others can authenticate data, or digital signatures, which unambiguously identify an author or source.¹⁸⁶ New companies are being formed to allow secure financial transactions on the Internet.¹⁸⁷ Expertise from a combination of such companies could be harnessed to provide the type of secure, imbedded certification described here.

B. A Comparison to Traditional Insurance

Most insurance schemes are based on traditional tort causation analysis.¹⁸⁸ A brief review of the various types of traditional insurance reveals their ultimate dependence on classic proximate cause analysis. These schemes' reliance on proximate causation is the fundamental reason why the Registry cannot function like and therefore is not an insurance scheme in the traditional sense.

Insurance usually just redirects, or spreads the risk of, an award of damages for a tort. Similarly, doctrines of vicarious liability such as master-servant and parent-child implicate identical issues of tort liability, the only difference being that the person paying is not the negligent actor.

Thus third-party (or liability) insurance pays out when someone else establishes a tort against the insured;¹⁸⁹ workers' compensation pays out when injury occurs in the workplace, when otherwise the employer might have been held to be responsible.¹⁹⁰ First-party insurance covers losses sustained directly by the insured.¹⁹¹

Some insurance policies are "all risk," and broadly insure against all losses described in the policy unless specifically excluded.¹⁹² Broadly speaking, the Turing Registry would offer a form of insurance, as it would "indemnif[y] another against loss."¹⁹³ But traditional causation, including its proximate cause analysis, is a fundamental underpinning of insurance coverage schemes.

Thus, for example, most policies will *not* cover losses caused by the "willful act of the insured."¹⁹⁴ While there is debate on the meaning of this willfulness exclusion, it has barred coverage for losses caused by the insured's intentional crimes,¹⁹⁵ for wrongful termination,¹⁹⁶ and for various "intentional" business torts.¹⁹⁷ As long as insurance companies can invoke exclusions such as these, the classic issues posed by proximate cause analysis will exist. More generally, even all risk, first-party insurance will have excluded perils or excluded risks, which will generate conflict between insureds and insurance companies on whether the excluded risk was or

was not the "proximate cause."¹⁹⁸ Thus both traditional third-party liability policies and first-party policies assume an intact, fully-functioning proximate cause doctrine. Neither of these insurance systems can address the liability of distributed AI.

C. The Registry's Limitations

There are at least two practical problems with the sketch provided above of the Registry's operations, and perhaps a third theoretical problem. First, the Registry will not preempt lawsuits directed at the scope of coverage provided by the Registry or lawsuits by individuals outside the system. The second problem is created by the technical difficulties in checking the reliability of agents. The third problem deals with the perceived difficulty of identifying the source of an agent or its code which causes damage in fact. A discussion of these will illuminate the proposed operation of a certification registry, and some of its limits.

1. Coverage Disputes and Victims Outside the System

First, there is some question whether the model sketched out above will really remove from the legal system disputes attendant on damage done by AIs. There is presently much litigation involving the scope of ordinary insurance coverage, and the model presented above would not necessarily preempt such lawsuits among (i) the Registry, (ii) programmers of agents and (iii) parties operating site hosts (such as United Airlines and hotels in the example above) who allow in certified agents.

Generally, though, these will be relatively ordinary contract disputes, not cases requiring an impossible cause-and-effect analysis of the damage done by intelligent agent ensembles. Thus the usual sort of contract (policy) lawsuit might be brought by damaged sites if the Registry failed to pay. Programmers might allege illegal restraints of trade if the Registry declined to certify certain agents. A site owner damaged by an agent designed to cause the damage could still sue the programmer of that agent.

Furthermore, the Turing system sketched out here may not have the ability to handle damage done when, for example, A1ef preempts a power supply and browns out the neighborhood, or when aircraft fly too low, creating a nuisance in the area. Distributed AI systems are likely to have consequences *outside* our computer networks, in physical contexts that cannot be "certified" by a Registry. But this issue, too, should be of decreasing importance over time. If the assumptions of this article are correct, complex digital systems will increasingly ramify throughout the infrastructure.¹⁹⁹ Thus, A1ef's assumed control of the power grid might well be a function of the AI's authorized reception by a machine host. If so, damages caused "in fact" by that infiltration would be compensable.

Unresolved by this discussion is the extent of the Registry's reimbursement coverage. There is a spectrum of potential damage ranging from direct injury, such as data corruption, to the immediate effects of a misjudgment, such as mid-air collision, to less direct effects such as neighborhood brown-outs, to highly indirect and consequential effects, such as a decline in the stock market value of a company that makes aircraft or radar parts. Presumably the Registry could, by contract, specify that compensation would be paid only for immediate pathological effects, i.e., for damage caused to networks and mechanisms immediately controlled by networks. Registries might also simply limit their payout on a per agent, per site or per event basis.

2. Technical Difficulties in Checking Software

There are problems with the technical methods to be used to check or analyze intelligent agents. We should assume the problems in verifying the actions of tomorrow's agents, and of ensembles of agents, to be more difficult than those involved in today's complex software environments.²⁰⁰ However, no Registry-Turing or its competitors-needs to establish the eternal reliability of agents. To the contrary, this article assumes that the agent ensembles are not always reliable. The issue is rather whether an examination of agents may reveal the *likelihood* of unpredictable and dangerous behavior over the long run. That is a technical question beyond the scope of this article, but success with that sort of stochastic evaluation-perhaps with the criteria I suggest above such as reactivity and environmental safeguards-is not inconsistent with the theoretical difficulties in predicting the specific behavior of complex programs.

3. Replicant Sampling and the revenge of the polymorph.

:10000000800900077470652E41534D188820000048

:10001000001C547572626F20417373656D626C656C²⁰¹

TGGAAGGGCTAATTCACCTCCCAACGAAGACAAGA²⁰²

A third problem associated with the Registry system is ensuring that a host system can identify certified agents. If AIs are polymorphic, and indeed if (as expected) we would expect their behavior will be *increasingly* polymorphic, will it be possible to identify these agents sufficiently to allow Turing certification?

The image that comes to mind is nailing Jell-O to the wall. There appears to be no guarantee that the specific code selected for certification will be involved in a pathological episode. Programs can replicate and send portions of themselves to distributed sites for processing; even localized programs can and will modify themselves. Thus we have the problem of *identifying* a "certified" agent. What does it mean to suggest that "Program Alef," or indeed any constituent agent, has been certified? How will we recognize Alef in a week, or in two years? Will we recognize Alef in a small codelet operating at a remote site? The problem arises because "copies" of digital information are perfectly indistinguishable from the "original." An inspection of code or data will not always reveal whether it has been truncated and therefore could have come from a different or an uncertified agent. The difficulty arises because Turing certifications apply to only a *portion* of the processing environment: only to polymorphic agents.²⁰³

It is essential to trace the behavior of a "processing environment" if any form of Turing certification is to succeed. Instead of looking to mutating processing nodes and executable files, though, we should look to the communications channels *between* such nodes. The emphasis remains on the overall processing environment, but focuses on a direct examination of information flow, and *not* on the nodes from which the flow originates or ends. Messages among processing nodes would be "tagged" with the appropriate authenticating codes, and Turing agencies would certify only programs that require the perpetuation of an electronic tag on all messages. Under the proposed model, no message is accepted or processed without the electronic tag. That tag - in effect the encrypted Turing certification - acts as a unique digital thumbprint, a coded stand-in for the program that assures the host processing environment of the bona fides of the messenger data (or instruction).

The model I propose to deal with these problems is based on the behavior of viruses.²⁰⁴ Mutating viruses will change small portions of their RNA (or DNA) sequence, but retain a persisting, distinct sequence otherwise. The small genetic change is enough to produce important differences in viral appearance and behavior, enough, for example, to avoid the effects of a host's immune defenses searching for the former viral incarnation. So too, certain structures can be locked into digital code without conflicting with the behavior of polymorphic programs. To borrow language from genetics, we can assure ourselves of some genetic persistence even where the phenotype is untrackably polymorphic. Intelligent programs, as such, may be sufficiently dispersed and mutable as to defy firm identification; but the messages passed by the programs and their codelets can be inspected for certification. Ultimately it is those messages, that information flow, which must be subject to Turing oversight.

The image, then, is of many gatekeepers watching over the flow of traffic. Only on presentation of an authentic "pass" are messages admitted to remote sites, or the value of variables passed to other codelets, subroutines or peripherals. The gatekeepers do not care where the message came from, or whether it is affiliated in some fashion with a Program Alef (whatever that is), as long as it is certified. The overhead of such a system might be high, but that is irrelevant.²⁰⁵ The point here is simply to model a target for Turing certification that does not depend on permanently fixing the boundaries of an artificial intelligence (or other program). And that can be accomplished. Turing Registries offer the possibility of a technological response to the quandary first posed by technology.

VI. Perspectives

There is no reason to suppose machines have any limitations not shared by man.²⁰⁶

It was a machine's dilemma, this inability to distinguish between actual and programmed experience.²⁰⁷

Effective, distributed AI presents the problem of judgment. As we rely on our electronic doppelgangers, we increasingly lose the referents of history, honest memory and truth as measured by correspondence with the way the world really is. Simulation, including computerized simulation, carries us away from these into what Baudrillard calls the "catastrophe of reality."²⁰⁸ We forget that digital space follows rules different from those of physical reality; we forget that digital space has no dimension and is ungoverned by the rules of physics, that it is just data.²⁰⁹ The logic of the simulacrum does not necessarily correlate with the physics of the real world. Simulacra may be simply self-referential, in the way that television shows just quote other shows and movies, and news stories cover the making of a film or "expose" the truth about another news show. A multimedia hyperlink between President Kennedy's death and the CIA does not mean that the two are in fact related. The logic of program flow does not explain how physical objects interact, and digital models may not accurately mimic physical reality.²¹⁰

Describing the 1989 missile attack on the U.S.S. Stark, author Gary Chapman recalls that the ship's defense systems probably failed because onboard computers classified the French-made-but Iraqi-launched-missiles as *friendly*. The programmers' assumptions did not correspond with the reality of the Persian Gulf conflict. Chapman writes:

[W]hen we talk about what goes on in a computer, we're talking about an entire complex of relations, assumptions, actions, intentions, design, error, too, as well as the results, and so on. A computer is a device that allows us to put cognitive models into operational form. But cognitive models are fictions, artificial constructs that correspond more or less to what happens in the world. When they don't, though, the results range from the absurd to the tragic.²¹¹

This disassociation from the real suggests that computer programs may at once be faithful to a digital logic and fail to exercise ordinary judgment. The problem is not new to those who develop artificial intelligence programs. Some researchers assemble enormous databases of facts and lessons; others draft heuristics, rules of thumb, in an effort to provide a reservoir of "common sense" on which programs may draw. Such efforts are probably doomed: they cast us back to the dark days of the rule-bound, domain-bound expert systems briefly described above.²¹² In any event, these are enormous undertakings, and cannot hope to underpin the distributed intelligences which we may use in the next decade. Other researchers may seek to impose "global controls": constraints on the judgments AIs may make, limits on the data they may consider and bars to the analogies they may devise. These are *all* attempts to substitute human judgment, formed in advance and so without the key facts, for the judgment of the AI created to handle the situation. AIs again confront us with the old, old issue of how much autonomy to allow the machine.

AIs' potential lack of common-sense sensitivity to the constraints of the physical world presents a serious risk to operations entrusted to autonomous artificial intelligences. But doubtless we will use intelligent systems, and pay the price, just as we use automobiles and pay the terrible toll of some 29,000 highway deaths a year.²¹³ Users and site operators can be circumspect, and should carefully screen agents, just as care is needed on the road. But AIs will inevitably arrive at "very strange answers"; when they do, the courts will not provide an effective forum.

As they assess the liability for AIs gone awry, courts will be tempted to use the same sort of proximate cause analysis they always have. But it will not work. We may know that AIs are involved as one of an infinite number of causes in fact. But against the background of ephemeral, distributed, polymorphic processing elements, judges will not be able to pluck out specific program applets, or human agencies, as proximate causes.

There is a risk when courts fail to provide a meaningful remedy for the felt insults of technology. Laws may be enacted and cases decided that make technological development too much of a litigation risk. Where social policy cannot find its target, liability may be imposed on the undeserving, and others may unjustly escape. The Turing Registry may provide a technological option to resolve the legal conundrum.

† 1996 Curtis E.A. Karnow.

† A.B. Harvard University, J.D. University of Pennsylvania. Mr. Karnow is a former federal prosecutor, and is currently a partner in a San Francisco law firm specializing in intellectual property and computer law. A summary version of this paper was delivered at DEFCON III, Las Vegas on August 5, 1995. The author may be contacted at "karnow@cup.portal.com".

1. DOUGLAS HOFSTADTER & MELANIE MITCHELL, *FLUID CONCEPTS AND CREATIVE ANALOGIES* 236 (1995).
2. JEAN BAUDRILLARD, *THE TRANSPARENCY OF EVIL* 53 (1993).
3. *See, e.g.*, Toshinori Munakata, *Commercial and Industrial AI*, COMM. ACM, March 1994, at 23 (manufacturing, consumer products, finance, management and medicine); Lawrence Gold, *If AI Ran The Zoo*, BYTE, Dec. 1995, at 79 (manufacturing in chemical and petrochemical industries).
4. This article does not address issues relating to consciousness, or programs' intentions; that is, whether computing systems can "think," "intend" or have purpose, or have a mind. *See generally* THINKING COMPUTERS AND VIRTUAL PERSONS (Eric Dietrich ed., 1994). The works of HOFSTADTER, *supra* note 1, and Dennett together suggest that "consciousness" is not sui generis and might emerge from a sufficiently complex and creative intelligence. DANIEL C. DENNETT, *CONSCIOUSNESS EXPLAINED* 431-40 (1991). Others vociferously disagree. JOHN R. SEARLE, *THE REDISCOVERY OF THE MIND* (1992). Occasionally, perceived disagreements

in this area stem from the ambiguous use of the term "intelligence." Recently the achievements of "Deep Blue," a chess-playing program, have spawned discussion on the meaning of intelligence in the machine context. *See* Bruce Weber, *A Mean Chess-Playing Computer Tears at the Meaning of Thought*, N.Y. TIMES, Feb. 19, 1996, at A1, A11 (interviewees variously ascribed the label "intelligence" to purely brute-force approaches to solving complex problems such as certain concrete chess problems, to accurate positional judgment in chess not a function of brute-force calculation, and to the ability to feel emotions or write music). This article uses the term "intelligence" as set out *infra* note 36 & part II.B.

5. *See generally* SHOSHANA ZUBOFF, *IN THE AGE OF THE SMART MACHINE* (1988).

6. For example, the Center for Nonlinear Studies at the Los Alamos National Laboratory in New Mexico investigates physical complexity in the behavior of fluids, gases and granular materials such as sand as well as in mathematical representations. Contact "office@cns.lanl.gov". These technically complex systems can evidence chaos, turbulence and emergent pattern formation. *See generally* JAMES GLEICK, *CHAOS: MAKING A NEW SCIENCE* (1987).

7. *The Computer Industry*, THE ECONOMIST, Sept. 17, 1994, at 20 (Survey).

8. Mark Weiser, *Some Computer Science Issues In Ubiquitous Computing*, COMM. ACM, July 1993, at 75, 75. James Gleick has written an amusing item describing the proliferation of very small computerized devices that we append to our bodies. James Gleick, *Watch This Space*, N. Y. TIMES MAG., July 9, 1995, at 14.

9. This total includes Apple computers, Intel-based computers (83% of the 1993 total) and others. ECONOMIST, *supra* note 7, at 4, 10. Worldwide sales of PCs may swell from \$95 billion in 1994 and \$116 billion in 1995 to \$185 billion in 1999. *World PC Surge Seen*, N. Y. TIMES, July 3, 1995, at 39.

10. Personal communications from Intel on file with author. The intermediate P6 chip has about 5.5 million transistors. *Twentieth Anniversary Report*, BYTE, Sept. 1995, at 74, 74.

11. Jeff Prorise, *Windows 95 Secrets*, 14 PC MAG., Dec. 19, 1995, at 247.

12. *See generally* ZUBOFF, *supra* note 5; Rob Kling et al., *Massively Parallel Computing and Information Capitalism, in A NEW ERA IN COMPUTATION* 191 (ed. N. Metropolis and Gian-Carlo Rota 1993). *See also* references to the confluence of computer and other systems, *infra* note 23 (convergence of telephony and other systems).

13. *See* Robert Orfali et al., *Intergalactic Client/Server Computing*, BYTE, Apr. 1995, at 108. Orfali projects "exponential network growth" integrating, for every user, many hundreds of programs and distributed data from around the world. *Id.*; *see also infra* note 15.

14. ECONOMIST, *supra* note 7, at 16.

15. *Regulating Cyberspace*, 268 SCIENCE 628 (1995). *See also* "URL ftp.misc.sr.com/pub/zone" on the Internet.

16. Jane Richter, *Distributing Data*, BYTE, June 1994, at 139.

17. The now-standard networked client/server environment consists of many scores of interacting program modules, quite aside from the switching hardware which acts as a program in its own right. Those modules may include system management applications such as configuration managers, storage managers, software distributors, license managers, print managers, etc.; network management frameworks such as Hewlett Packard's OpenView or IBM's Netview; perhaps hundreds of client applications such as word processors, spreadsheets, drawing programs, desktop publishing and communications; client operating systems such as Windows, OS/2 Warp and others; network operating systems such as Windows NT Server, Vines, Appleshare and others; server operating systems which may or may not include the client operating systems; server applications such as Sybase SQL Server, Oracle 7 and dozens of others; and so on. Each of these programs contains its own program and data modules, some but not all of which are shared with other programs. *See generally* *ComputerWorld*, CLIENT/SERVER J., Aug. 1995.

18. Anne Knowles, *InterAp Assigns Intelligent Agents to the Web*, PC WEEK, June 12, 1995, at 42, 47 (quoting Bob Johnson of

Dataquest Inc.). *See also* Nick Wingfield, *Internet Apps to Get Intelligent Search Agents*, INFOWORLD, May 15, 1995, at 16. These agents are discussed at greater length *infra* part II.D.

19. *See infra* part II.A.

20. *See, e.g.*, Douglas B. Lenat, *CYC: A Large-Scale Investment in Knowledge Infrastructure*, COMM. ACM, Nov. 1995, at 33, 33.

21. Another example is to provide the pilot of a complex, high-performance aircraft "with enhanced situational awareness by sorting and prioritizing data, analyzing sensor and aircraft system data, distilling the data into relevant information, and managing the presentation of that information to the pilot." B. Chaib-draa, *Industrial Applications of Distributed AI*, COMM. ACM, Nov. 1995, at 49, 49-50.

22. *See generally* M. K. El-Najdawi & Anthony C. Stylianou, *Expert Support Systems: Integrating AI Technologies*, COMM. ACM, Dec. 1993, at 55, 56.

23. For a more formal list of what we might expect from these programs, *see* MAUREEN CAUDILL & CHARLES BUTLER, *NATURALLY INTELLIGENT SYSTEMS* 152, 153 (1990) ("The Characteristics of Learning Systems").

24. For convenience, I suggest here a single "program." As noted below, a more accurate notion is an *ensemble of programs* acting in conjunction, constituting a processing environment. The inability of the traditional tort system to evaluate the actions of these ensembles results from the fact that they are indeed ensembles with disparate etiologies, not lone programs authored by a single person or company and residing at a single physical location.

25. *See* Julia King & Thomas Hoffman, *Hotel Heading for 'Net Without Reservations*, COMPUTERWORLD, Nov. 27, 1995, at 1. Some hotels permit Internet access to their reservations systems and are now moving toward the integration of these networks, with computerized property-management systems in charge of group sales and catering, remote check-in and check-out, credit-card authorization and settlement, food and beverage management, and database marketing. *Id.* at 28.

26. *See infra* part IV.A.

27. The confluence of voice and data transmission will lead to a greater integration of computers and telephony, eventually eviscerating the distinction between these systems. *See generally* Neal Weinberg, *Computer/phone Finds Its Voice*, COMPUTERWORLD, Dec. 18, 1995, at 55; *Cool Today, Hot Tomorrow*, BYTE, Sept. 1995, at 84; *Collision!*, BYTE, Sept., 1995, at 199. Note also the expected merger of the television set and the computer. *All-in-one Computers: Computer-TV Hybrids Invade the Den*, BYTE, Sept. 1995, at 32.

The recent enactment of the Telecommunications Act of 1996, signed by President Clinton on February 8, 1996, will permit companies in the cable, television and telephone industries to compete in all these related arenas. Pub. L. No. 104, 110 Stat. 56 (1996). Already television cable is being used to provide Internet access, and some commentators foresee an expansion of that technological merger. Sarah L. Roberts, *The Internet Comes To Cable*, PC MAG., Dec. 5, 1995, at 31.

28. *See generally* LAUREN RUTH WIENER, *DIGITAL WOES* (1993); more details are available from the Internet USENET group "comp.risks" at "URL <http://catless.ncl.ac.uk/Risks>" and archived at "URL <ftp:unix.sri.com/risks>". RISKS-LIST: RISKS-FORUM DIGEST is also available at this address. The unavoidable risks posed by complex software are discussed further *infra* part II.C.

29. *Distributed* intelligence suggests a series of programs physically resident at various disparate sites, interacting with each other to present the user with a single integrated result. *See, e.g.*, Andy Reinhart, *The Network With Smarts*, BYTE, Oct. 1994, at 51.

30. *Brown v. United States*, 790 F.2d 199 (1st Cir. 1986) (holding defendants not liable for drowning of fisherman in storm which National Weather Service did not predict).

31. *Cf. Wenninger v. United States*, 234 F. Supp. 499 (D. Del. 1964).

32. *See, e.g.*, *Summers v. Tice*, 199 P.2d 1 (Cal. 1948).

33. We are already familiar with an incipient version of this. Users of home PCs are routinely told their problems have to do with the complex interaction of CD-ROMs, operating systems, memory management software, IRQ and other settings within the machine, various peripherals, sound cards, graphical user interfaces and other applications, and on and on.

34. Users might be aware of, and indeed command, higher order decisions such as "if the temperature is above 43 degrees then close the circuit." But virtually all program decisions are of a lower order, such as "if certain address in memory contains the datum XX then replace it with datum ZZ," or "if a datum at memory location is XX then jump to a different section of this program and execute the instruction found there."

35. CAUDILL & BUTLER, *supra* note 23, at 25 (1990).

36. The wonderful and nebulous term *artificial intelligence* covers a plethora of programming techniques and goals. Direct modeling of the activity of the brain's network of neurons is one such field; also included are *case-based* reasoning systems, which try to apply rules to new factual scenarios. AI systems are used to model complex movement in robotics, including the difficult areas of perception and visual and aural discrimination. Munakata, *supra* note 3, at 23, 24-25. *See also* Gold, *supra* note 3 at 79 (describing hybrid AI systems using combination of neural nets and other programming techniques to control complex manufacturing processes). As this article notes below, the term "intelligent" in the context of so-called intelligent agents refers to agents (program chunks or "modules") that can communicate and work with other agents to produce a larger program, which in turn appears to mimic the higher cognitive abilities of humans. *See infra* part II.D. (text accompanying note 68).

37. Here is how one team of researchers introduced the idea of neural nets:

[T]he most common models take the neuron as the basic processing unit. Each such processing unit is characterized by an activity level (representing the state of polarization of a neuron), an output value (representing the firing rate of the neuron), a set of input connections, (representing synapses on the cell and its dendrite), a bias value (representing an internal resting level of the neuron) and a set of output connections (representing a neuron's axonal projections) Thus, each connection has an associated weight (synaptic strength) which determines the effect of the incoming input on the activation level of the unit. The weights may be positive (excitatory) or negative (inhibitory). Frequently, the input lines are assumed to sum linearly yielding an activation value.

David Rumelhart et al., *The Basic Ideas in Neural Networks*, COMM. ACM, Mar. 1994, at 87.

38. *Id.* at 89.

39. The physical appearance of a neural net need be little different from that of any computer. It may have a camera, for example, to enable the input of digitized video. The nodes are simply values (if that is not too loose a term) embodied in software. For a discussion of neural nets generally, see the "Frequently Asked Questions" (FAQ), *available at* "URL ftp:rtfm.mit.edu/pub/usenet/news.answers" (filename "neural-net-faq"). *See also* "URL http://wwwipd.ira.uka.de/~prechelt/

FAQ/neural-net-faq.html" on the World Wide Web. There is a good deal of literature on the subject of neural nets. *See generally* CAUDILL & BUTLER, *supra* note 23; *see also* Bill Machrone, *Care and Feeding of Neural Nets*, PC WEEK, June 5, 1995, at 63. Neural nets can be used to analyze highly complex groups of constraints: for example, motion controls for robots that require the analysis of feedback from the environment present difficult nonlinear control issues, solvable by such nets. Haruhiko Asada, *Representation and Learning of Nonlinear Compliance Using Neural Nets*, 9 IEEE TRANSACTIONS ON ROBOTICS & AUTOMATION 863 (Dec. 1993). *See also* Gary S. May, *Manufacturing ICs the Neural Way*, IEEE SPECTRUM, Sept. 1994, at 47 (describing how the many factors that go into the calculation of efficient chip fabrication can be solved with these expert nets).

40. May, *supra* note 39, at 47.

41. CAUDILL & BUTLER, *supra* note 23, at 241-60 (1990).

42. Such systems include what professor Jim Bezdek terms a basic "computational neural network." These give the appearance of intelligence, but they operate "solely on numerical data such as that obtained from sensors and computational pattern recognition . . . produc[ing] numerical results that reveal structure in sensor data." *Technology Focus*, COMPUTER DESIGN, Sept. 1994, at 74. The ability to manage substantial combinatorial complexity, though, is not the exhibition of intelligence or true learning. "This idea that neural networks learn is just stupid," Bezdek notes. "They don't learn An algorithm is just an algorithm." *Id.*

43. HOFSTADTER & MITCHELL, *supra* note 1.
44. *Id.* at 97.
45. *Id.* at 124-25.
46. CAUDILL & BUTLER, *supra* note 23, at 155.
47. HOFSTADTER & MITCHELL, *supra* note 1, at 256.
48. HOFSTADTER & MITCHELL, *supra* note 1, at 91 (describing parallel architecture by which bottom-up and top-down processing influence each other). *See* G. Hinton et al., *The 'Wake-Sleep' Algorithm for Unsupervised Neural Networks*, 268 *SCIENCE* 1158 (1995) (describing a computer neural network by which bottom-up "recognition" connections convert input vectors to representations in one or more hidden layers of the neural net. The top-down "generative" connections are then used to reconstruct an approximation of the input vectors from the underlying representations).
49. MARVIN MINSKY, *THE SOCIETY OF MIND* (1985); *A Conversation with Marvin Minsky About Agents*, *COMM. ACM*, July 1994, at 23, 23. *See generally* DENNETT, *supra* note 4.
50. HOFSTADTER & MITCHELL, *supra* note 1, at 215.
51. *Id.* at 235.
52. *Id.*
53. MARK A. LUDWIG, *COMPUTERS, VIRUSES, ARTIFICIAL LIFE AND EVOLUTION* 86 (1993).
54. "Autonomy" is a relative term. Some systems are more or less autonomous than others, and there is a broad range of autonomy and automation. THOMAS B. SHERIDAN, *TELEROBOTICS, AUTOMATION, AND HUMAN SUPERVISORY CONTROL* 356-60 (1992). I have not expressly treated that range in this paper. I have addressed the issue instead through the prism of multiple concurrent causation, which recognizes variable contributions of humans and machines to a given result. *See infra* part IV.
55. Buggy software is released even when known to be buggy. Corel was told of crashes in its drawing program, but released it anyway, apparently to meet certain market deadlines. Pardhu Vadlamudi, *Corel Faces Buggy Software Backlash*, *INFOWORLD*, Oct. 23, 1995, at 25.
56. *See generally* Bev Littlewood & Lorenzo Strigini, *The Risks of Software*, *SCI. AM.*, Nov. 1992, at 62.
57. Manuel Blum & Sampath Kannan, *Designing Programs That Check Their Work*, 42 *J. ASS'N COMPUTING MACH.* 267 (1995).
58. Bev Littlewood & Lorenzo Strigini, *Validation of Ultrahigh Dependability for Software-based Systems*, *COMM. ACM*, Nov. 1993, at 69.
59. *Id.* at 78-79.
60. WIENER, *supra* note 28, at 96-98.
61. *Id.* at 96.
62. The number rises from 20 (1 command) to 400 (2 commands) to 8000 (3 commands) and finally to 10,240,000,000,000 (10 commands). *Id.*
63. Tom Davey, *Chip Makers Split on Listing Bugs*, *PC WEEK*, Dec. 18, 1995, at 115 (noting errors in initial batch of Intel's Pentium

Pro (known in development as the P6) which could corrupt data, as well as recalling problems with the original Pentium chip). The first releases of the Pentium chip (originally known as the P5) in late 1994 contained a FPU (Floating Point Unit, or arithmetic coprocessor) which was comprised of registers unable to correctly handle a divide function, the so-called FDIV error. *See Editorial*, BYTE, Sept. 1995, at 126 (report of Pentium bug).

64. The Federal Aviation Administration's Advanced Automation System's millions of lines of code are buggy, and so the nation's skies remain under the control of decades-old computer technology. When three lines of code were changed in a telecommunications program in the summer of 1991, California and much of the Eastern seaboard lost phone service. Two cancer patients were killed in 1986 by overdoses of radiation from a computer-controlled radiation therapy machine; errors included a so-called "race" condition (where two or more threads in a parallel processing system do not execute as predicted). Alan Joch, *How Software Doesn't Work*, BYTE, Dec. 1995, at 49-50.

65. *Id.*

66. *See generally Intelligent Agents*, COMM. ACM, July 1994 (special issue); *Software Agents Prepare to Sift the Riches of Cyberspace*, 265 SCIENCE 882 (1994) [hereinafter *Software Agents*]; INTELLIGENT AGENTS: THEORIES, ARCHITECTURES AND LANGUAGES (1995); "URL <http://www.doc.mmu.ac.uk/STAFF/mike/atal95.html>" on the World Wide Web; "URL <http://www.cs.umbc.edu/agents>" on the World Wide Web. *See also* Deiri Masaru, *Knowbotics: A Talk on the Wild Side with Pattie Maes*, INTERCOMMUNICATIONS, Annual 1994, at 110.

A recent study suggests (perhaps with a bit of exaggeration) that "[a]gents will be the most important computing paradigm in the next ten years." BIS STRATEGIC DECISIONS, PRAGMATIC APPLICATION OF INFORMATION AGENTS (1995).

67. *See* Andy Reinhardt, *The Network With Smarts*, BYTE, Oct. 1994, at 51.

68. HOTT (Hot Off The Tree), Apr. 25, 1994 (Internet download) carried a report describing a program known as

Hoover, from Sandpoint Corporation (Cambridge, MA), an information-gathering program. Hoover's search results are compiled into a customized electronic newsletter, with headlines that can be clicked on with a mouse to retrieve full-text articles. Microsoft's Office suite includes Intelligence for real-time spelling error correction . . . Other software packages include Beyondmail from Beyond, Inc. and Open Sesame! from Charles River Analytic (Cambridge, MA). Beyondmail automates responses to incoming e-mail. Open Sesame! monitors repetitive PC activity . . . and then, in essence, automatically creates intelligent, autonomous macros.

Id. AT&T planned to introduce Telescript e-mail in 1994, which would have allowed users to type in the addressee's phone number; an agent would then look up the e-mail address corresponding to that number and deliver the message to the addressee's computer. General Magic plans to license Telescript freely to other companies. Presumably the system will allow cross-platform, network-independent messaging, insulating users and programmers from the complexities of network protocols. Tom R. Halfhill & Andy Reinhardt, *Just like Magic?*, BYTE, Feb. 1994, at 22; Michael Fitzgerald, *Agent Technology Stirs Hope of Magical Future*, COMPUTERWORLD, Jan. 31, 1994, at 37; Yvonne L. Lee, *Telescript Eases Cross-network Communication*, INFOWORLD, Jan. 17, 1994, at 22; *Agents of Change*, BYTE, Mar. 1995, at 95.

69. Peter Wayner, *Agents Away*, BYTE, May 1994, at 113.

70. *See* Edmund Durfee & Jeffery Rosenschein, *Distributed Problem Solving And Multi-Agent Systems: Comparisons and Examples*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE 52 (1994) (Proceedings of the 13th International Distributed Artificial Intelligence Workshop, published by the American Association for Artificial Intelligence) [hereinafter DISTRIBUTED ARTIFICIAL INTELLIGENCE]; Ernest Edmonds et al., *Support for Collaborative Design: Agents and Emergence*, COMM. ACM, July 1994, at 41; *Software Agents*, *supra* note 66, at 883.

71. Masaru, *supra* note 66, at 114-15. *See* JyiShane Liu & Katia Sycara, *Distributed Problem Solving Through Coordination in a Society of Agents*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 169.

72. Bernardo A. Huberman, *Towards A Social Mind*, 2 STAN. HUMAN. REV. 103, 107 (1992) (*expanded in* HUBERMAN, ORIGINS OF THE HUMAN BRAIN 250 (1995)). *See also* Bernardo A. Huberman & Tad Hogg, *Distributed Computation as an Economic System*, J. ECON. PERSP., Winter 1995, at 141. I am grateful to Mr. Huberman, who is a Research Fellow with Xerox PARC in Palo

Alto, California, for providing me with his articles.

73. Frederick Hayes-Roth & Neil Jacobstein, *The State of Knowledge-Based Systems*, COMM. ACM, Mar. 1994, at 27, 35.

74. *Id.* at 38.

75. See generally John Markoff, *Staking Claim in Alternative Software on the Internet*, N.Y. TIMES, Sept. 25, 1995, at D4; Gus Venditto, *Java: It's Hot, But Is It Ready To Serve?*, INTERNET WORLD, Feb. 1996, at 76, 78.

76. As discussed *infra* note 101, these distributable agglutinations of data and program code are known as *objects*. The notion of combined data and programming code is embodied in another term used by Java programmers, "executable content." JOHN DECEMBER, PRESENTING JAVA 6 (1995).

77. TIM RITCHEY, JAVA! 15-16 (1996).

78. *Id.* at 23-24.

79. *Beyond Java: Distributed Objects on Web*, PC WEEK, Dec. 18, 1995, at 48.

80. Stuart J. Johnston & Kim S. Nash, *Capitulation!*, COMPUTERWORLD, Dec. 11, 1995, at 1.

81. Java is fast and reportedly highly secure. Recognizing the risks posed by allowing applets-true blue executable programs-to download from remote sites into user's host machines, one title blandly assures the reader that "No Java applet is able to steal information or damage your computer in any way." ARTHUR VAN HOFF ET AL., HOOKED ON JAVA 17 (1996). Security is achieved by (i) restricting the environment in which Java applets can run (a notion treated under the rubric of "global controls" in this article) and (ii) verifying the individually transmitted "bytecodes" which are subsequently interpreted, and then run together on the user's host machine (approximating the certification process discussed in this article). *Id.* at 17-18; RITCHEY *supra* note 77, at 50-51, 99. However, more recently, security flaws have been discovered. *Netscape Flaw Could Cause Harm*, MARIN INDEP. J., Feb. 22, 1996, at C3; see also Gary Anthes, *Still a Few Chinks in Java's Armor*, COMPUTERWORLD, Feb. 19, 1996. Details and further discussion may be found in RISKS-LIST: RISKS-FORUM DIGEST, vols. 17.83 and 17.85, available from the Internet USENET group "comp.risks" at "URL <http://catless.ncl.ac.uk/Risks>".

82. Reinhardt, *supra* note 67, at 51, 64 (TeleScript security).

83. Wayner, *supra* note 69, at 116. Wayner does explain that the version of agent software he examines (TeleScript) has a "security" system by which the host computer can allow only previously authorized users to send in agents. For more on the security issues associated with these and related agents, see *supra* note 81.

84. *Housebroken Virus*, INFOSECURITY NEWS, Sept./Oct. 1994. Neither my use of the term "virus" nor that of *InfoSecurity* is precise here. Normally machine viruses are considered to be *self-replicating*, and not necessarily unintentionally (or indeed intentionally) destructive. See generally ROGUE PROGRAMS: VIRUSES, WORMS AND TROJAN HORSES (Lance J. Hoffman ed., 1990); MARK A. LUDWIG, THE LITTLE BLACK BOOK OF COMPUTER VIRUSES (1991). However, the classic definition and my use share the connotation of a program that is (i) hidden within (and dependent upon) a software host and (ii) independent, mobile and transfers itself from host to host.

85. See generally Curtis Karnow, *Recombinant Culture: Crime In The Digital Network*, available at "URL http://www.cpsr.org/cpsr/computer_crime/net.crime.karnow.txt" on the World Wide Web. See also Jim Louderback, *A Virus by Another Name Causes Equal Pain*, PC WEEK, Apr. 3, 1995, at 106. Louderback writes that TeleScript is not a virus, "but it makes it easier for rogue programs to spread." *Id.* at 106.

86. Other popular programs such as Borland's Quattro Pro spreadsheet and WordPerfect have similar macro capability.

87. See, e.g., Jason Pontin, *Macro Virus Threat Continues*, INFOWORLD, Dec. 11, 1995, at 36; see also Gary Anthes, *Macro Viruses Pose Hazard to PC Health*, COMPUTERWORLD, Feb. 19, 1996, at 45.

88. Pontin, *supra* note 87, at 36 (quoting Karen Black, Symantec Corp). The macro viruses appear to be a reincarnation of a problem that cropped up a few years ago in an early version of Microsoft's object linking and embedding (OLE) technology. Certain applications such as Microsoft's Office suite of programs allowed users to create, in effect, self-executing data files which would be held inside another data file (such as letter or a chapter of a book). For example, a document might have a graphic "object" (say a picture of an apple) embedded as an illustration in the document; OLE would *execute* the object, automatically running a new program to display it, as the reader read the document. But OLE "may inadvertently create a backdoor through which malicious individuals can enter to embed destructive commands, viruses and worms in any number of applications that are distributed via electronic mail or network protocols." Michael Vizard, *Security Woes Dull OLE Luster*, COMPUTERWORLD, Dec. 6, 1993, at 1.
89. Huberman, *supra* note 72, at 107. The use of what Huberman calls "global controls" would reduce the autonomy of the system and undermine its claim to collaborative, emergent intelligence. *See infra* part IV.A.2.b.
90. *In Short*, INFO. WK., May 22, 1995, at 88, *cited in* RISKS-LIST: Risks-Forum Digest vol. 17.13, available from the Internet USENET group "comp.risks" at "URL <http://catless.ncl.ac.uk/Risks>". *See also* *Win95 Wizard Leads to Confusion*, INFOWORLD, May 29, 1995, at 6.
91. Microsoft's position is available at "URL ftp.microsoft.com/peropsys/win_news/regwiz.txt" on the Internet.
92. *See, e.g.*, CAL. CIV. CODE § 3426 (West Supp. 1996) (Uniform Trade Secrets Act).
93. 17 U.S.C. §§ 501-10 (1977).
94. *See generally* Laurence H. Tribe, *Rights of Privacy and Personhood*, in AMERICAN CONSTITUTIONAL LAW § 15 (1988) (right to control personal information).
95. 18 U.S.C.A. §§ 2510-22 (West 1970 & Supp. 1995). *See* CAL. PENAL CODE § 502 (West Supp. 1996) (discussing unauthorized access to computer data and systems). For more information on so-called web-wandering "robots," "spiders" and other programs which among other things take data from remote locations without permission., *The Web-Crawler Wars*, 269 SCIENCE 1355 (1995) (discussing the dangers of software "robots" accessing information without human supervision and judgment); *see generally* Jeff Frentzen, *Spiders, Worms, and Robots Crawl in the Web*, PC WEEK, Feb. 13, 1995 (*citing* "URL <http://web.nexor.co.uk/mak/doc/robots/robots.html>" on the World Wide Web).
96. *See generally* JOHN HOLLAND, ADAPTATION IN NATURAL AND ARTIFICIAL SYSTEMS (1992). Professor Holland and others have experimented with systems that compel competition between programs to evaluate their suitability for certain tasks. They then use selected parts of successful programs for "genetic" recombination, creating succeeding generations of programs. These new programs again compete and begin the loop again. After many iterations, highly successful programs can be generated. *Id.* at 3-4.
97. *See infra* note 104.
98. These drivers are small programs "designed to handle a particular peripheral device such as a magnetic disk or tape unit." THE NEW HACKER'S DICTIONARY 134 (Eric Raymond ed., 1991).
99. "The most important event in the history of software happened somewhere around 1959, when the designers of a programming language called 'Algol 60' realized that *you can build a large program out of smaller programs.*" DAVID GELERNTER, MIRROR WORLDS 54 (1992). *See supra* note 17 (listing multiple modules used in current client-server environment).
100. LEE ATKINSON & MARK ATKINSON, USING BORLAND C++ 432 (1991).
101. A similar effect can be simulated within the Microsoft operating system Windows. An "object" such as a spreadsheet or graphic can be created and then inserted into data created by another program, such as into a letter created by a word processor. This so-called "object" acts as a combination of the data (i.e., the spread-sheet values the numbers, or the picture) *and* the program required to modify the data (the spread-sheet program, or the paint program). *See supra* note 88 (OLE corruption problems).
102. Christine Comaford, *Inside the Black Box Of Objects*, PC WEEK, Nov. 20, 1995, at 18.

103. John Perry Barlow, *The Economy of Ideas*, WIRED, Mar. 1994, at 84, 84.

104. For example, companies such as Compuserve and Netscape, as well as Windows' manufacturer Microsoft, sell Windows-compatible Internet communications programs. Their programs call on a library of functions—a set of small programs or modules—found in (among other places) a dynamic link library (DLL) drafted to a public technical standard known as the Windows Sockets. This library of modules is called WINSOCK.DLL. Multiple programs can call on the functions contained in WINSOCK.DLL. But when the user installs Microsoft's Internet package, the installation quietly renames WINSOCK.DLL to WINSOCK.OLD, and then copies its *own* WINSOCK.DLL to the user's drive. Programs other than Microsoft's can not use this new DLL, and so they fail. Brian Livingston, *How Microsoft Disables Rivals' Internet Software*, INFOWORLD, Sept. 25, 1995, at 42. As of last Fall, at least, Microsoft had not told its competitors how to use Microsoft's superseding WINSOCK.DLL. *Id.*

105. *See infra* note 203.

106. C.S. LEWIS, THE SCREWTAPE LETTERS XXI.

107. Other elements include a "duty" to the injured party and "damage" or injury caused by a violation of that duty. The element of "duty" can encompass a host of subsidiary issues. *See generally* 5 BERNARD WITKIN, SUMMARY OF CALIFORNIA LAW § 3 *et. seq.* (1988). Confusingly, the notion of "duty" is often used to test whether a cause of injury should or should not be the "legal cause" or legally responsible cause for that injury. This issue is properly discussed *infra* in part III.A & III.B in the context of foreseeable risk and multiple concurrent causation. But the "duty" element is not always coterminous with the scope of the foreseeable risk; in those circumstances, the existence of a duty is a separate element needed for liability. *See* 3 FOWLER V. HARPER ET AL., THE LAW OF TORTS § 18.8 (1986). Only the essential element of causation is discussed in this paper.

108. Maybe. *See Williams v. Saga Enters., Inc.*, 274 Cal. Rptr. 3d 901 (Cal. Ct. App. 1990).

109. Sometimes, yes. *See Jackson v. Ryder Truck Rentals, Inc.*, 20 Cal. Rptr. 2d 913, 920 (Cal. Ct. App. 1993).

110. Yes. *People v. Caldwell*, 681 P.2d 274 (Cal. 1984). This recalls an earlier case in which the defendant shot his brother-in-law. *People v. Lewis*, 57 P. 470 (Cal. 1899) (the victim, realizing that he would die a long, slow and very painful death, slit his own throat; the defendant was convicted of manslaughter for that death). *See also People v. Gardner*, 43 Cal. Rptr. 2d 603 (Cal. Ct. App. 1995) (compiling cases on "derivative liability for homicide").

111. Perhaps both. *See Summers v. Tice*, 199 P.2d 1 (Cal. 1948).

112. Any and all of these persons *may* be legally liable. *Mitchell v. Gonzales*, 274 Cal. Rptr. 541 (1990), *aff'd*, 819 P.2d 872 (Cal. 1991). This is so even though any one of the causes alone would never have resulted in the drowning. *See generally* *Mitchell v. Gonzales*, 819 P.2d 872 (Cal. 1991); *Morgan v. Stubblefield*, 493 P.2d 465 (Cal. 1972); *Lareau v. S. Pac. Transp. Co.*, 118 Cal. Rptr. 837 (Cal. Ct. App. 1975) (car and train combined to kill passenger); *DaFonte v. Up-Right, Inc.*, 828 P.2d 140 (Cal. 1992) (combination of failure to warn, insufficient supervision, operator negligence and poor design of equipment all led to serious injury to a 15-year-old's hand as he cleaned a moving conveyor belt of mechanical grape harvester).

113. *Lineaweaver v. Plant Insulation Co.*, 37 Cal. Rptr. 2d 902 (1995) (asbestos).

114. *Compare* *Merrill v. Los Angeles Gas & Elec. Co.*, 111 P. 534 (Cal. 1910) (escaping gas and light caused explosion) *with* *Lowenschuss v. S. Cal. Gas Co.*, 14 Cal. Rptr. 2d 59 (Cal. Ct. App. 1992) (gas company has no responsibility to purge gas from pipes and house meters in the path of on coming fire).

115. 4 FOWLER V. HARPER ET AL., THE LAW OF TORTS § 14.3 (1986).

116. *Id.* § 20.2.

117. 199 P.2d 1 (Cal. 1948).

118. *See, e.g., Lineaweaver v. Plant Insulation Co.*, 37 Cal. Rptr. 2d 902 (Cal. Ct. App. 1995) (*Summers* doctrine will not hold multitude of asbestos manufacturers responsible when an *undetermined* one was responsible.).

119. *Vigiolto v. Johns-Manville Corp.*, 643 F. Supp. 1454, 1457 (W.D. Pa. 1986). This switch of burdens is reminiscent of the venerable doctrine of *res ipsa loquitur*, which holds that when all of the many possible instrumentalities of injury are under the control of defendants, it is up to the defendants to prove their own innocence, rather than to the plaintiff to prove defendants' culpability. *See, e.g., Cooper v. Horn*, 448 S.E.2d 403, 405 (Va. 1994) (discussing *res ipsa loquitur*).
120. *Lineaweaver*, 37 Cal. Rptr. 2d at 906-08 (Cal. Ct. App. 1995).
121. 607 P.2d 924 (Cal. 1980).
122. *Id.* at 611-12.
123. *Mullen v. Armstrong World Indus.*, 246 Cal. Rptr. 32, 35 n.6 (Cal. Ct. App 1988) (asbestos not "fungible," so *Sindell* does not apply).
124. *Id.*; *Vigiolto v. Johns-Manville Corp.*, 543 F. Supp. 1454, 1460-61 (W.D. Pa. 1986); *In re Related Asbestos Cases*, 543 F. Supp. 1152, 1158 (N.D. Cal. 1982). *See generally* RESTATEMENT (SECOND) OF TORTS § 433 B (3) (1965).
125. *See generally* *Maupin v. Widling*, 237 Cal. Rptr. 521 (Cal. Ct. App. 1987); HARPER, *supra* note 115 § 20.4 *et seq.*
126. RESTATEMENT (SECOND) OF TORTS § 431 cmt. a (1965). *See* *People v. M.S.*, 896 P.2d 1365, 1386-87 (Cal. 1995) (Kennard, J., concurring); *Mitchell v. Gonzales*, 819 P.2d 872, 882-85 (Cal. 1991) (Kennard, J., dissenting) (referring to the "social evaluative process" involved in deciding which of the infinite "causes" will be targeted for legal liability); HARPER, *supra* note 115 § 20.4 ("Policy considerations underlie the doctrine of proximate cause.").
127. *See generally* WILLIAM L. PROSSER, THE LAW OF TORTS 270 (1971).
128. *See, e.g.,* HARPER, *supra* note 115 § 20.5 (foreseeability as part of the analysis underpinning "legal" cause). While this article discusses only tort liability, it is worth noting that reasonable foreseeability also plays an important role in traditional contract law, in that breaching defendants normally are liable only for damages which were reasonably foreseeable. *Hadley v. Baxendale*, 9 Ex. 341 (1854); *Martin v. U-Haul Co. of Fresno*, 251 Cal. Rptr. 17, 23-24 (Cal. Ct. App. 1988).
129. *Cooper v. Horn*, 448 S.E.2d 403 (Va. 1994).
130. *O'Brien v. B.L.C. Ins. Co.*, 768 S.W.2d 64, 68 (Mo. 1989); *see also* *Doe v. Manheimer*, 563 A.2d 699 (Conn. 1989) (rapist's conduct deemed not reasonably foreseeable misconduct); *Erikson v. Curtis Inv. Co.*, 447 N.W.2d 165 (Minn. 1989) (a parking ramp operator owes customers some duty of care to protect against foreseeable criminal activity); *Akins v. D.C.*, 526 A.2d 933, 935 (D.C. 1987) (defendant computer manufacturer, whose negligence allowed a computer error that resulted in a criminal's early release from prison, held not liable for remotely foreseeable criminal assault).
131. *Prashant Enter. v. State*, 614 N.Y.S.2d 653 (1994); *Saden v. Kirby*, 660 So. 2d 423 (La. 1995).
132. *Boyd v. Washington-St. Tammany Elec. Coop.*, 618 So. 2d 982 (Ala. Civ. App. 1993).
133. *Knapp v. Neb. Pub. Power Dist.*, No. A-93-134 1995 WL 595691 (Neb. App. Oct. 10, 1995).
134. *Bradford v. Universal Constr. Co.*, 644 So. 2d 864 (Ala. 1994).
135. *Mann v. Anderson*, 426 S.E.2d 583 (Ga. Ct. App. 1992).
136. The language in many of these cases mixes the test for causation *in fact* with that for *proximate cause*. Thus the courts' preliminary statements of the legal principles are often a muddle, although the actual evaluations and results in these cases are not necessarily wrong. For example, *Knapp* states that the defendant is liable "unless the sole proximate cause of that damage is an 'extraordinary force of nature.'" 1995 WL 595691 at *3. But this begs the analysis: the natural force (here, a rainstorm) will be held to

be "solely" responsible-i.e. will be the "sole proximate cause"-if the natural force was so extraordinary that the humans could not reasonably have foreseen it. Later, the court does in fact use that foreseeability analysis. *Id.* at *4 (citing *Cover v. Platte Valley Pub. Power & Irrigation Dist.*, 75 N.W.2d 661 (1956)). *Cooper* also states that the Act of God doctrine applies only when the natural force "was the sole proximate cause of the injury . . . [and] all human agency is to be excluded from creating or entering into the cause of the mischief, in order that it may be deemed an Act of God." *Cooper v. Horn*, 448 S.E.2d. at 425 (citing *City of Portsmouth v. Culpepper*, 64 S.E.2d 799, 801 (Va. 1951)). Having found a human agency in the flood damage (a human-built dam that collapsed), *Cooper* decided that the Act of God doctrine was inapplicable. *Id.* That analysis is flawed because there is always human agency as at least one of the contributing causes in fact. (The court was, however, probably influenced by its view that the human actions contributed to the damage were indeed foreseeable.) Human "agency" should be excluded as a consequence, not predicate, of proximate cause analysis. In contrast to *Cooper*, in *Mann* the Act of God doctrine (which was also defined as excluding "all idea of human agency," 426 S.E.2d at 584) was held potentially applicable (thus relieving humans of all liability) to a series of automobile collisions in the fog-where the human agencies were absolutely obvious. But again, *Mann* correctly focused on the foreseeability of the ultimate injuries in the context of the fog and other weather conditions as presented to the defendants at the time of the accident. *Id.* at 585.

137. *Pamela B. v. Hayden*, 31 Cal. Rptr. 2d 147, 160 (Cal. Ct. App. 1994) (holding landlord liable for failing to provide adequate security in the underground garage of his apartment building where a tenant was raped).

138. *Sprecher v. Adamson Co.*, 636 P.2d 1121 (Cal. 1981) (Bird, C.J.).

139. Former President Jimmy Carter reported to the White House "over an unsecured radio link" on his way back from a peace mission to Haiti, and expressed surprise that his conversation had been recorded by others. Mark Lewyn, *Eavesdroppers Speak Softly and Carry a Big Scrambler*, BUS. WK., Oct. 3, 1994 at 6, 6 (quoting CNN interview with J. Carter). Some years ago, Prince Charles too apparently did not know that his cellular phone conversations with a mistress were public. *Id.* See generally *Rosh v. Cave Imaging Sys., Inc.*, 32 Cal. Rptr. 2d 136, 141-42 (Cal. Ct. App. 1994).

140. HARPER, *supra* note 115 § 20.5 at 167 (citing James, *The Qualities of The Reasonable Man In Negligence Cases*, 16 MO. L. REV. 1, 5-15 (1951)).

141. *Pamela B. v. Hayden*, 31 Cal. Rptr. 2d 147, 160 (Cal. Ct. App. 1994).

142. TERRY WINOGRAD & FERNANDO FLORES, UNDERSTANDING COMPUTERS AND COGNITION 155 (1986).

143. SHERIDAN, *supra* note 54, at 341.

144. WINOGRAD & FLORES, *supra* note 142, at 155-56.

145. *Id.*

146. Hayes-Roth & Jacobstein, *supra* note 73, at 27, 38.

147. In the electronic arena, humans and software communicate in the same way, and when both are capable of spawning software agents, it is likely that others in the electronic medium will be unable to distinguish between human- and machine-originated communications. See generally Pattie Maes, *Artificial Life Meets Entertainment: Lifelike Autonomous Agents*, COMM. ACM, Nov. 1995, at 108. Consistent with the general model of intelligent agents mapped out in this article, Maes describes agents:

[Agents are] distributed, decentralized systems consisting of small competence modules. Each competence module is an "expert" at achieving a particular small, task-oriented competence. There is no central reasoner, nor any central internal model. The modules interface with one other via extremely simple messages As a result the behavior produced is robust, adaptive to changes, fast and reactive.

Id. at 111.

148. See generally DECEMBER, *supra* note 76.

149. Last year, software agents were "evolved" by other computer programs in an apparently successful effort to create intelligent

autonomous agents that would assist in securing computer systems from unauthorized attacks. *Security Schemes Aspire to No-Fuss System Protection*, 270 SCIENCE 1113, 1114 (Nov. 1995).

150. ED KROL, THE WHOLE INTERNET USER'S GUIDE & CATALOG 47-48 (1992).

151. ARCHIE, VERONICA and GOPHER are three euphoniously named search engines that reside at various systems throughout the world, ready to accept requests from a series of other remote programs (or, lest we forget, from humans as well) to search the Internet for selected items.

152. See Durfee & Rosenshein, *supra* note 70, at 54. Air traffic control presents a series of inherently concurrent computational problems, and thus it is well suited to the use of a large collection of concurrently active agents. *Id.* at vii. Of course, a wide variety of uses exists for intelligent agents in other distributed systems, such as for information management, Michael Huhns et. al., *Global Information Management via Local Autonomous Agents*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 120; Riyaz Sikora & Michael J. Shaw, *Manufacturing Information Coordination and System Integration by a Multi-Agent Framework*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 314; manufacturing, H. Van Dyke Parunak, *Deploying Autonomous Agents on the Shop Floor: A Preliminary Report*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 259; air combat and related situational awareness contexts, Anand S. Rao & Graerne Murray, *Multi-Agent Mental-State Recognition and its Application to Air-Combat Modeling*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 264; and telecommunications, Robert Weihmayer & Hugo Velthuisen, *Application of Distributed AI and Cooperative Problem Solving to Telecommunications*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 353.

153. Of course, the ensemble of intelligent agents contemplated by this article is not operational, and so the outline here is speculative.

154. Professor Sheridan has a useful list describing many of the precursors to the automated systems listed below. He describes modern autopilots and associated programming functions, on-board collision-avoidance systems and the like. SHERIDAN, *supra* note 54, at 239-45 (1992). Many sources, such as advanced pilot's manuals, provide information on the human and machine systems currently controlling the nation's airspace. See, e.g., RICHARD TAYLOR, INSTRUMENT FLYING (1978); JEPPESON SANDERSON, ADVANCED PILOT MANUAL (1981).

155. That is, the physical movement of machines linked to Alef.

156. See Jörg P. Müller & Markus Pischel, *Integrating Agent Interaction into a Planner-Reactor Architecture*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70 at 232.

157. T. Finin et. al., *KQML-A Language and Protocol for Knowledge and Information Exchange*, in DISTRIBUTED ARTIFICIAL INTELLIGENCE, *supra* note 70, at 99 (discussing protocol for exchanging information and knowledge).

158. Huhns et. al., *supra* note 152, at 128.

159. An elementary example of this used today is the "programming" by pilots of their transponders, devices that respond to usually ground-based radar queries with a code selected by the pilot. Some codes indicate emergencies; some mean the aircraft's radio is out of commission; others indicate an aircraft flying under visual (as opposed to instrument) flight rules. Pilots also individually "program" or set their altimeters, information from which is passed via radio to ground-based controllers, and at times to other aircraft to determine if a collision is imminent. Pilot-created programs also navigate via selected way points at given altitudes, and so on.

160. HOFSTADTER & MITCHELL, *supra* note 1, at 235 (1995).

161. See generally HOFSTADTER & MITCHELL, *supra* note 1.

162. Huberman, *supra* note 72, at 107.

163. See SHERIDAN, *supra* note 54, at 356-60 (1992) (discussing supervisory control of automation).

164. See Pattie Maes, *supra* note 147, at 108.

165. *See supra* part II.C. (discussing the unreliability of software).

166. PROSSER & KEETON, *THE LAW OF TORTS* § 43 at 293, 297-99 (1984); 4 F. HARPER, ET AL., *THE LAW OF TORTS*, § 20.5 at 164-65 (1988) (results not extraordinary for liability to attach); RESTATEMENT (SECOND) OF TORTS § 281 cmts. e & f, 435(2), 451 (1965). *See also id.* §§ 450, 451 (no liability for acts of God if those acts are extraordinary and bring about an injury different in kind from that originally threatened by defendant's act). *See generally supra* note 136 (discussing Acts of God).

167. HARPER, *supra* note 115 § 20.5 at 162.

168. *See Lopez v. McDonald's*, 238 Cal. Rptr. 436, 445-46 (Cal. Ct. App. 1987) (holding McDonald's not liable for deaths of plaintiffs caused by an unforeseeable mass murder assault at its restaurant).

169. *Compare State v. White*, 660 So. 2d 664 (Fla. 1995) (police department's failure to update its computer records for days after an arrest warrant was served caused second unjustifiable arrest; court suppressed evidence seized at second arrest) *with Arizona v. Evans*, 115 S. Ct. 1185 (1995) (no suppression of evidence when computer was negligently maintained by court officer *outside* of police department). The suppression rule is designed to encourage the police to act properly: the Florida Supreme Court argued that policy is furthered by suppression in the Florida case, and not in the Arizona matter. *See also Bank Leumi Trust Co. v. Bank of Mid-Jersey*, 499 F. Supp. 1023 (D.N.J. 1980) (bank liable for computer's failure to read handwritten notation on a check; not an Act of God).

170. Courts do not actually require that the person held liable have undertaken this forecast. Rather, the courts hold that if it was reasonable to have expected that forecast, the defendant will be treated as if the forecast had been made. *See supra* note 141.

171. *See I. Trotter Hardy, The Proper Legal Regime For 'Cyberspace,'* 55 U. PITT. L. REV. 993, 1013, 1040 (1994).

172. Brooke Crothers & Bob Fracis, *Flawed IDE Controller Corrupts Data*, INFOWORLD, August 14, 1995, at 6. The USENET group "comp.risks" has reported on this example of unpredictable conflicts erupting between operating systems and symbiotic hardware components:

Intel and other computer companies are trying to determine the extent of problems caused by a flaw in an RZ-1000 EIDE controller chip that is included in some early PCI motherboards manufactured by PC Tech. The flaw was discovered in 1994 and was corrected through a software 'patch,' but the latest version of [IBM's operating system] OS/2 disables that patch.

RISK-LIST: RISKS-FORUM DIGEST, Aug. 18, 1995, available from the Internet USENET group "comp.risks" at "URL <http://catless.ncl.ac.uk/Risks>" (citing INVESTOR'S BUS. DAILY, August 16, 1995, at A15).

173. John Rheinfrank, *A Conversation with John Seely Brown*, 11 INTERACTIONS 43, 50 (1995) (quoting John Brown, Chief Scientist at Xerox's PARC).

174. That is the case now, but I have previously outlined a framework for treating electronic personalities as fully fledged legal entities. Curtis Karnow, *The Encrypted Self: Fleshing Out The Rights of Electronic Personalities*, 13 J. COMPUTER & INFO. L. 1 (1994). That framework does not, however, address or solve the problems discussed here of distributed agency and the breakdown of causation.

175. *See supra* part I.A.

176. JEAN BAUDRILLARD, *THE ILLUSION OF THE END* 40 (1994).

177. HOFSTADTER & MITCHELL, *supra* note 1, at 236.

178. Apostles of the master will recognize the source. WILLIAM GIBSON, *NEUROMANCER* (1984). Alan M. Turing was a brilliant British mathematician who devised in 1936 the formal architecture of the computer, an abstraction termed a "Turing Machine." DAVID HAREL, *THE SCIENCE OF COMPUTING* 202 (1987). The "Turing test" is one putatively for intelligence, perhaps conscious intelligence, in which conversational responses of hidden humans and machines are passed via console to human testers. If the testers cannot distinguish the human from the machine, the machine is said to pass the Turing test. There is little agreement on whether the

test is meaningful or indeed has ever been passed. *See, e.g.,* Paul Churchland et al., *Could A Machine Think? in THINKING COMPUTERS & VIRTUAL PERSONS*, *supra* note 4, at 157-158; SHERIDAN, *supra* note 54, at 348-49 .

179. SHERIDAN, *supra* note 54; *see also* part IV.A.2.b. (pathological operations).

180. Programs such as Phil Zimmermann's Pretty Good Privacy(r) [PGP] can do this now. (A disclaimer: the author represents Mr. Zimmermann.) Encryption software can provide the means to authenticate a digital message as from a given source. PGP is available as freeware. *See* PHILIP ZIMMERMANN, *THE OFFICIAL PGP USER'S GUIDE* (1995); Max Schireson, *Decoding the Complexities of Cryptography*, *PC WEEK*, January 10, 1994, at 84. *Cf.* BRUCE SCHNEIER, *APPLIED CRYPTOGRAPHY* (1994); John Perry Barlow, *A Plain Text on Crypto Policy*, 36 *COMM. ACM* Nov. 1993, at 11, 21; Steven Levy, *Crypto Rebels*, *WIRED*, May-June 1993, at 54; A. Michael Froomkin, *The Metaphor Is the Key: Cryptography, the Clipper Chip, and the Constitution*, 143 *U. PA. L. REV.* 709 (1995); Jonathan Erickson, *Cryptography Fires Up the Feds*, *DR. DOBB'S JOURNAL*, Dec. 1993, at 6; Brian Hayes, *The Electronic Palimpsest*, *THE SCIENCES*, Sept.-Oct. 1993, at 10. *See also* Curtis Karnow, *ENCRYPTION & Export Laws: The Algorithm as Nuclear Weapon*, *SOFTWARE PUBLISHER*, Nov.-Dec. 1994.

181. Agents may mutate, and accordingly the certification must be able in some sense to "follow" the agent through its incarnations. This issue is discussed *infra* part V.C.3.

182. Hotels do in fact link their systems to the Internet. *See supra* note 23.

183. *See generally* DURFEE & ROSENSCHEIN, *DISTRIBUTED ARTIFICIAL INTELLIGENCE*, *supra* note 70. In what I have dubbed a *processing environment* it is not possible to separate environment from programs. *See also supra* part II.F.

184. *See supra* part I.V.A.2.b. Limiting the access, reactivity and (in effect) the scope of an agent's judgment doubtless will limit the harm agents can do. However, this will come at the cost of eviscerating agents' ability function as distributed intelligences-e.g., their ability to manage the complex electronic province entrusted to them.

185. *Corporations Eye Private Security Systems*, *BYTE*, August 1995, at 36.

186. Nick Wingfield, *Digital IDs to Help Secure Internet: Certifying Authorities to Promote Electronic Commerce*, *INFOWORLD*, October 23, 1995, at 12.

187. CyberCash Inc. enables encrypted transmissions of credit card information across the Internet. Elizabeth Corcoran, *Reston's CyberCash Joins Internet Companies' Rush to Wall St.*, *WASH. POST*, Dec. 23, 1995, at C1.

188. *See* ROBERT E. KEETON, *INSURANCE LAW, BASIC TEXT* 318 (1971) ("The analogy between insurance and tort cases on issues of proximate cause is quite close.").

189. *See, e.g.,* *Fireman's Fund Inc. Co. v. City of Turlock*, 216 Cal. Rptr. 796 (Cal. Ct. App. 1985); *Int'l. Surplus Kines Ins. Co. v. Devonshire Coverage Corp.*, 155 Cal. Rptr. 870 (Cal. Ct. App. 1979).

190. CAL. INS. CODE § 109 (West 1993) ("[I]nsurance against loss from liability imposed by law upon employers to compensate employees and their dependents for injury sustained by the employee arising out of and in the course of employment, irrespective of negligence or of the fault of either party.").

191. *Garvey v. State Farm Fire & Casualty Co.*, 770 P.2d 704, 705 n.2 (Cal. 1989) (contrasting first and third-party policies).

192. *Strubble v. United Servs. Auto. Ass'n*, 110 Cal. Rptr. 828, 830-32 (Cal. Ct. App. 1973).

193. *See* CAL. INS. CODE §§ 22, 250 (West 1993) (defining "insurance").

194. CAL. INS. CODE § 533 (West 1993) (forbidding coverage for willful acts by the insured). *See also* CAL. CIV. CODE § 1668; *Clemmer v. Hartford Ins. Co.*, 587 P.2d 1098, 1105-06 (Cal. 1978).

195. *See, e.g.*, *State Farm Fire & Casualty Co. v. Dominguez*, 182 Cal. Rptr. 109 (Cal. Ct. App. 1982) (first degree murder).

196. *See, e.g.*, *St. Paul Fire & Marine Ins. Co. v. Superior Court*, 208 Cal. Rptr. 5 (Cal. Ct. App. 1984).

197. *See, e.g.*, *Aetna Casualty & Sur. Co. v. Centennial Inc. Co.*, 838 F.2d 346 (9th Cir. 1988); *State Farm Fire & Casualty Co. v. Geary*, 699 F. Supp. 756 (N.D. Cal. 1989); *Allstate Ins. Co. v. Interbank Fin. Servs.*, 264 Cal. Rptr. 25 (Cal. Ct. App. 1989).

198. *Garvey v. State Farm Fire & Casualty Co.*, 770 P.2d 704 (Cal. 1989). *See also LaBato v. State Farm Fire & Casualty Co.*, 263 Cal. Rptr. 382 (Cal. Ct. App. 1989). *Garvey* used the phrase "efficient proximate cause" in the sense of the "predominating cause" as distinguished from some "initial" or "moving" causes which would give an undue emphasis on a temporary prime or "triggering" causation. *Garvey*, 770 P.2d at 707. Thus *Garvey* applies proximate cause in roughly the traditional way, as outlined *supra* part III. *See generally* CAL. INS. CODE §§ 530, 532 (West 1993) (causation requirements for coverage).

199. *See supra* part I.A.

200. These difficulties are discussed at length *supra* part II.C.

201. These are the first two lines of the hex listing for a modified *Trident Polymorphic Engine*, used to link into an otherwise monomorphic virus. The linked engine will cause the virus to mutate unpredictably and, depending on the virus search tools employed, unrecognizably. LUDWIG, *supra* note 53, at 356 (1993).

202. This commences the nucleotide sequence for the HIV retroviral provirus, responsible for the disease AIDS and associated with adult T-cell leukemia lymphoma. Lee Ratner et al., *Complete Nucleotide Sequence of the AIDS Virus, HTLV-III*, 313 NATURE 277 (1985). Even at this relatively early date the investigators recognized the apparent polymorphism of the virus. *Id.* at 283. We now know that HIV mutates "incredibly rapidly, often within the very person it infects; there have been instances of viruses in the same individual varying by as much as 30 percent." PETER RADETSKY, THE INVISIBLE INVADERS 341 (1991).

203. The problem of digital polymorphism underlies much of the current anxiety in intellectual property circles. Barlow, *The Economy of Ideas*, *supra* note 103, at 84; Pamela Samuelson, *The NII Intellectual Property Report*, 37 COMM. ACM, Dec. 1994, at 21. Copyright problems arise as programs, at various levels of abstraction, are sampled and appropriated by third parties. *See, e.g.*, Raymond T. Nimmer et al., *Software Copyright: Sliding Scales and Abstract Expression*, 32 HOUSTON L. REV. 317 (1995); Curtis Karnow, *Data Morphing: Ownership, Copyright & Creation*, 27 LEONARDO 117 (1994); Pamela Samuleson, *Digital Media and the Law*, 34 COMM. ACM, Oct. 1994, at 23. These problems are magnified when Internet access allows essentially invisible hypertext linking of widely distributed documents, a linking which eviscerates the integrity of previously discrete documents, texts and graphics. *See generally* MARY E. CARTER, ELECTRONIC HIGHWAY ROBBERY: AN ARTIST'S GUIDE TO COPYRIGHTS IN THE DIGITAL ERA (1996). Similar issues arise in the context of trademarks in the digital context. The essential distinctiveness of trademarks and trade dress is difficult to fix when morphing software subtly alters texts and design, the touchstones of trademark law, along a spectrum of infinite detail. And trademark law's utter reliance on the notion of "customer confusion" is subtly eroded when customers shop and buy products on line, instead of physically inspecting the products in the physical world. For the basics of trademark law, see J. THOMAS MCCARTHY, MCCARTHY ON TRADEMARKS AND UNFAIR COMPETITION (1995).

Digital sampling disbands that which previously had been a unity. Sampling erodes the underlying notion of property—a notion derived from bounded, tangible physical land—as the logical focus of the law. Barlow, *supra*, at 84. It remains unclear which notion, or underlying legal metaphor, will replace this type of property. This is not the place to detail these problems, but the reference here does confirm the futility of treating software as identifiable, localized, discrete property. Emerging problems in the digital copyright realm concomitantly suggest problems in the present context. Certifying software, as we might tag wild animals on the veldt for ecological analysis, will not work.

204. *See supra* part II.E.

205. The overhead cost may not in fact be prohibitive or otherwise technically difficult. After a draft of this paper was completed, a report appeared on the new programming language Java, discussed above at text accompanying note 79. In language reminiscent of my requirements for Turing certification of processing environments, Java constantly monitors foreign programs for viruses and other rogue programs, both while the foreign modules or "applets" (i.e., small program applications, roughly what I have termed "codelets") are resident, and after they have been terminated. John Markoff, *A Software Language to Put You In The Picture*, N. Y. TIMES, Sept. 25, 1995, at C1.

- 206 MARVIN L. MINSKY, *COMPUTATION: FINITE AND INFINITE MACHINES* vii (1967).
- 207 JAMES LUCENO, *THE BIG EMPTY* 82 (1993).
208. BAUDRILLARD, *supra* note 176, at 113.
209. Edmond Couchot, *Between the Real and The Virtual*, *INTERCOMMUNICATIONS*, Annual 1994, at 16.
210. *See generally* Curtis Karnow, *Information Loss and Implicit Error in Complex Modeling Machines*, NTT Media Lab, *Networked Realities '94* (Tokyo 1994); Wiener, *supra* note 28; Littlewood & Strigini, *supra* note 56.
211. Gary Chapman, *Making Sense Out Of Nonsense: Rescuing Reality from Virtual Reality*, in *CULTURE ON THE BRINK: IDEOLOGIES OF TECHNOLOGY* 149, 151 (Gretchen Bender & Timothy Druckrey eds., 1994).
212. *See supra* part II.A; *see generally* HOFSTADTER & MITCHELL, *supra* note 1, at 319 *et seq.*
213. *Highway Safety-Causes of Injury in Automobile Crashes*, GAO Rept. No. GAO/PEMD-95-4, May 9, 1995, at 1.