# Computers, visualization, and the nature of reasoning

Jon Barwise and John Etchemendy[1]

The computer is bringing about a revolution in our understanding of inference, representation, and reasoning, some of the most fundamental notions of logic. The revolution is far from complete, but we think the direction is clear enough. In this article we describe how the computer led the two of us first to change the way we teach elementary logic, and eventually to rethink basic assumptions about the subject matter of our discipline. We think the story is a remarkable case study of the synergy between teaching, technology, and research.

## 1 Some autobiography

Our story begins in 1983 when we both returned to Stanford from teaching at other universities. Elementary logic instruction is part of the bread-and-butter teaching in Stanford's philosophy department, as it is in most philosophy departments. There is a difference, though, because Stanford has a long tradition of using computers in logic instruction. This dates back to the late 1960's, when Patrick Suppes introduced his program *Valid* into Philosophy 57, Stanford's elementary logic course. This program ran on a mainframe computer devoted to nothing else, and provided students with an entire course of instruction in introductory logic. Like most logic courses offered from the 60's through the 80's, it focused on teaching the basic syntactic rules of proof in a formal calculus. Unlike most, it covered the subject in considerable depth, delving into axiomatic systems of some sophistication.

[1]The programs described in this paper were conceived by the authors, but would never have become realities without the design and programming talents of Gerry Allwein, Dave Barker-Plummer, Kalpana Bharadwaj, Alan Bush, Doug Felt, Dan Fish, Christopher Fuselier, Bill Graham, Mark Greaves, Adrian Klein, Mike Lenz, Steve Loving, Eric Ly, Atty Mullins, Pete Murray, Mark Ravaglia, Xin Wei Sha, Jason Strober, Rolf van Widenfelt, and Rick Wong. Financial support was received from Stanford University's Faculty Author Development Program and Center for the Study of Language and Information, and from Indiana University's Visual Inference Laboratory.

Over the years, thousands of Stanford students learned logic from *Valid*. While one section of Philosophy 57 was always "person taught"—for computer-phobic students—the bulk of elementary logic instruction was shouldered by Suppes' pioneering program. This freed most of the Stanford philosophy faculty from the burden of teaching elementary logic. In particular, the two of us could teach more advanced and interesting logic courses. That first year we taught courses in computability (Turing machines, recursive functions, undecidablity) and mathematical logic (truth, models, soundness, completeness).

In the winter of 1984, the same year we returned to Stanford, Apple Computer introduced its Macintosh. The graphical capabilities of the Macintosh immediately captured our imagination as a way to solve some basic pedagogical problems we had encountered in these logic courses. One problem in teaching computability had to do with Turing machines, a model of computation developed in the thirties by the famous logician Alan Turing, and a fundamental notion in theoretical computer science. The problem was the difficulty of giving students a real sense of the power of Turing machines. Since Turing machines get extremely complicated extremely fast, the only exercises we could assign (and have any hope of grading) were toy problems that called for the construction of very simple machines. And even with these simple machines students often failed to get them right, due to the difficulty of verifying that a particular machine did what it was supposed to do. For many students the gap between the problems they could hope to solve with pencil and paper and the evident power of the modern computer made many claims in the course seem far-fetched. For example, students were led to accept Church's Thesis, the claim that Turing machines can compute any function that can be calculated by mechanical or algorithmic means, more on the basis of our authority than on the basis of their own appreciation of the incredible power of these machines.

The problem in mathematical logic was of a very different sort. In our classes many students who had been quite successful in constructing proofs using *Valid*, students who should have understood the symbolic language in which those proofs were couched, in fact had great difficulty grasping the very intuitive notion of truth in a structure. This lack of understanding evidenced itself repeatedly. For example, students would make egregious errors in translating between sentences of English and sentences of first-order logic, errors that would have been inconceivable had they really understood the meanings of both sentences.

The Macintosh's graphical capabilities inspired us to tackle these problems in a new way. We envisioned tools that would facilitate the student's ability to visualize the abstract subject matter of logic, and thereby work more effectively with it. Over the next four years, funded by Stanford's Faculty

Author Development project, we worked with teams of student programmers to develop programs we called *Turing's World* [2] and *Tarski's World* [3]. As hoped, these programs have been extremely successful in addressing the pedagogical problems at which they were aimed. What we did not expect, however, was that they would lead us to rethink the conceptual foundations of our subject. In order to explain this second result, we need to briefly describe the programs.

## Turing's World

Introduced by Alan Turing in 1936, Turing machines are one of the key abstractions used in modern computability theory, the study of what computers can and cannot do. A Turing machine is a particularly simple model of the digital computer, one whose operations are limited to reading and writing symbols on a linear tape, or moving along the tape to the left or right. The tape is marked off into squares, each of which can be filled with at most one symbol. At any given point in its operation, the Turing machine can read or write on only one of these squares, the square located directly below its "read / write" head.

In Turing's World the tape is represented by a narrow window that sits at the bottom of the screen. Figure 1 shows the tape with a series of A's and B's written on it, and with the read / write head located on the leftmost of these symbols.

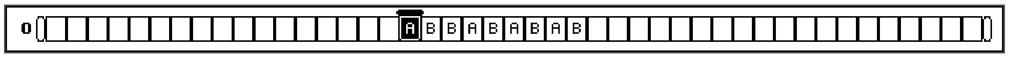


Figure 1: A tape from Turing's World.

A Turing machine has a finite number of states and is in exactly one of these states at any given time. Associated with these states are instructions telling the machine what action to perform if it is currently scanning a particular symbol, and what state to go into after performing this action. The states of a Turing machine are generally represented by a "flow" or "state" diagram, using circles for the states and labeled arcs for the instructions associated with those states.

For example, Figure 2 shows the state diagram of a Turing machine with just two states.

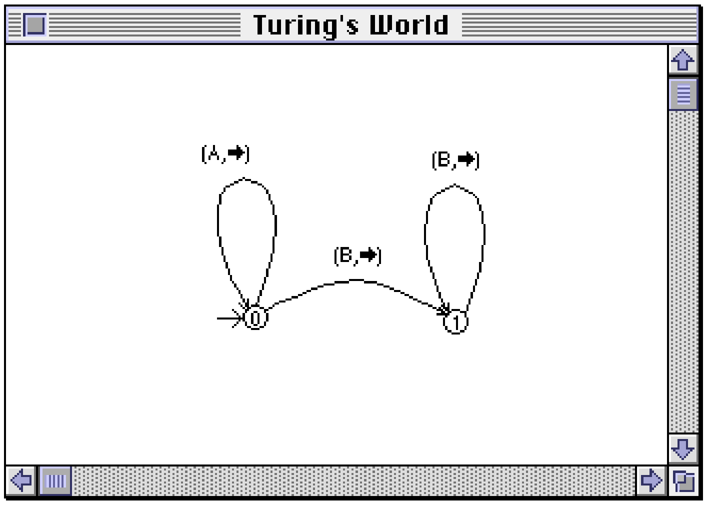


Figure 2: A simple Turing machine.

The linguistic description of this same machine would be given as follows:

In state 0: if you see an A, move right and go into state 0.
In state 0: if you see a B, move right and go into state 1.
In state 1: if you see a B, move right and go into state 1.

or, in abbreviated “4-tuple” notation:

0, A, R, 0
0, B, R, 1
1, B, R, 1

This machine will run down a string of A’s and B’s, stopping at the first A it sees after a B. For example, if run on the sample tape above, it would stop when it got to the fourth symbol from the left, because it would then be in state 1 with no instructions about what to do next.

In Turing’s World, a collection of graphical tools lets you design Turing machines by directly drawing their state diagrams. When you run a Turing machine in Turing’s World, the operation of the machine is displayed graphically, both on the tape and in the state diagram window. On the tape, the read / write head moves, making the changes required by the machine you’ve designed. In the state diagram, the nodes and arcs highlight to show the changing state of the computation. Turing’s World also allows students to display the text-based “4-tuple” description of their machines, though we have found that they rarely do.

Despite their simplicity, Turing machines can be designed to compute remarkably complex functions. In fact, they are generally thought to be as powerful (in theory) as any possible computer. Finding out what they can do is half the fun of studying computability. Figure 3 shows part of the detail of a Turing machine that adds two numbers expressed in decimal notation.

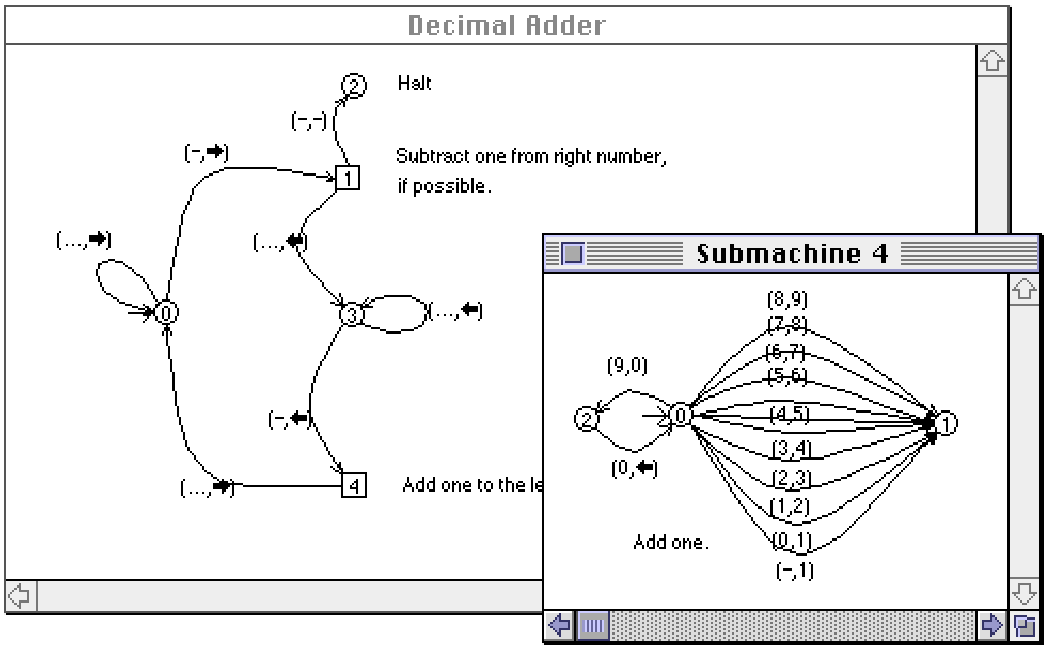
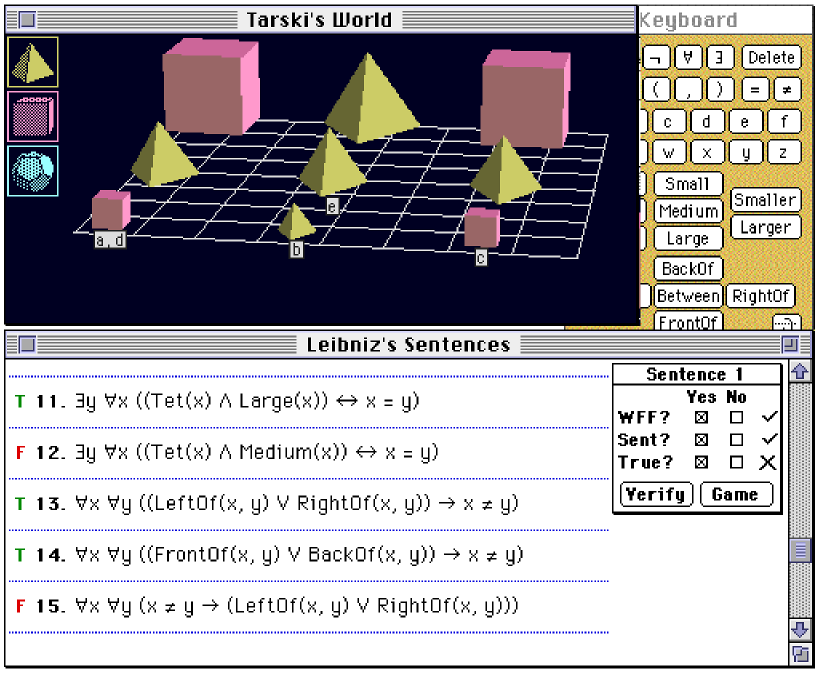


Figure 3: A Turing machine for adding in base 10.

The ability to design a Turing machine by simply drawing its state diagram, combined with the ability to run the machine and watch it go through its transitions, is a powerful aid in the student’s understanding of the intuitive idea of a Turing machine. By allowing students to introduce submachines (as illustrated in Figure 3), the program lets them quickly build up a powerful arsenal of Turing machines and to combine them in important ways. In our classes we now routinely ask students to design a Universal Turing machine, a machine that acts as a fully programmable computer. This exercise is several orders of magnitude more complex than what we could expect of our students before the advent of Turing’s World. Our students now learn to appreciate the power of Turing’s abstract machines, and have fun while doing it.

## Tarski’s World

The goal of Tarski’s World is to teach students the symbolic language at the core of modern, first-order logic. The program allows students to represent simple, three-dimensional worlds inhabited by geometric objects of various shapes and sizes, and to test first-order sentences to see whether they are true or false in those worlds. Figure 4 shows an example of such a world and some sentences (numbers 11–15 out of a longer list) that the student has been examining.



T 11. ∃y ∀x ((Tet(x) ∧ Large(x)) ↔ x = y)

F 12. ∃y ∀x ((Tet(x) ∧ Medium(x)) ↔ x = y)

T 13. ∀x ∀y ((LeftOf(x, y) ∨ RightOf(x, y)) → x ≠ y)

T 14. ∀x ∀y ((FrontOf(x, y) ∨ BackOf(x, y)) → x ≠ y)

F 15. ∀x ∀y (x ≠ y → (LeftOf(x, y) ∨ RightOf(x, y)))

Figure 4: An example from Tarski's World.

As you can see from the T's and F's at the left, three of the displayed sentences are true, while two are false. Sentence 12, for example, is false: it claims that there is a unique, medium-sized tetrahedron, when in fact there are three such blocks in the depicted world. If students do not understand why the sentence is false, they can choose to play an interactive game that successively breaks down the claim into simpler constituent parts. Though we cannot demonstrate the game in a static description, its rules are based on the meanings of the symbols (quantifiers and connectives) of the language, and thereby drive home the real import of the sentence at issue. The game continues until the student's misunderstanding of the sentence becomes completely evident. Thus the student quickly identifies and corrects his or her own misconstruals of the language, rather than waiting for classroom help or, as we found to be so common, getting through the entire course without the problem being recognized.

Despite its simplicity, there are many ways that we use Tarski's World in teaching the language of first-order logic. Most straightforwardly, we give students exercises in which they are asked to evaluate a set of sentences in a given world. But we also give them a set of sentences and ask them to construct a world that makes the sentences all true. Or we start with a world and ask the students to express certain facts about the world in the first-order language. Tarski's World can also be used to show various kinds of non-entailments. For instance, the example shown above proves that neither sentence 12 nor sentence 15 follows from 11, 13, and 14. Finally, it can be used for posing some interesting deduction problems, as we will see in a moment.

## 2 Visualization and reasoning

In using Turing's World in our teaching, we became enormously impressed by the cognitive power of the graphical representation of Turing machines over the more text-based, 4-tuple representation. There was no comparison between the two when it came to ease of design, understanding, or verification of Turing machines. While both modes of representation are available in Turing's World, students never resorted to the linguistic representation except when told to do so. This sparked our interest in the general topic of visual programming languages even before it become a lively topic in computer science.

We must confess, though, that we did not really grasp the revolutionary import for logic of what we were observing in our students. After all, we already knew that it was much easier to use the diagrammatic representation of a Turing machine: that is why we designed Turing's World with its graphical capabilities in the first place. The full import of what we had noticed only hit us with Tarski's World.

We mentioned earlier that in using Tarski's World, we assigned students exercises that required deductive reasoning for their solutions. In these problems, we would present a picture of a blocks world constructed in Tarski's World, as well as a list of sentences involving names, some of whose referents were not indicated in the picture. We would then ask students to deduce certain facts from the information they had been given.

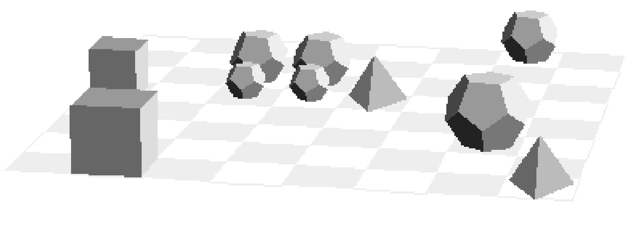


Figure 5: A blocks world.

### An example

For a simple example of this sort, consider the situation depicted in Figure 5. This figure shows us a world made up of blocks arranged on a chess board. The blocks come in three shapes (cubes, tetrahedra, and dodecahedra) and three sizes (small, medium, and large). Suppose our goal is to determine as much as we can about two blocks, named *b* and *c*. Ideally we would be able to identify them in the picture. Failing that, we should at least figure out as much as we can about them.

Suppose we are initially given the following information:

> Nothing is on a square adjacent to block *b* and block *c* is not a cube.

Clearly on the basis of this information we cannot identify either of the blocks. We can rule out some possibilities, though. For example, *c* cannot be either of the cubes on the left, and *b* cannot be in the central grouping of five blocks, since each of them is adjacent to others. Indeed, a moment's thought shows that there are eight possible blocks that could be *c* and five for *b*, giving us a total of forty cases consistent with this information.

Suppose we are next given the following information:

> Block *c* is either small or large.

This information tells us nothing more about *b*, of course, but it does significantly narrow the range of possibilities for *c*. Given our earlier information, we now see that there are two cases to consider: *c* has to be either one of the two small dodecahedra in the center, or the large dodecahedron toward the front. We cannot tell which one it is, but nonetheless we can observe in each case that *c* is a dodecahedron.

Continuing, suppose we are given the following piece of evidence:

> Block *b* is farther back than either of the tetrahedra.

This tells us that *b* must be one of the three backmost dodecahedra. But since we already know that *b* is not in the central group of blocks, we can conclude that *b* is the lone dodecahedron at the back of the board. We have thus successfully identified *b*.

Suppose our final piece of information is:

> Block *b* is larger than block *c*.

We already know that *c* is either one of the two small dodecahedra or else the large dodecahedron up front. Since we now know that *b* is a medium-sized block, we can rule out the possibility that *c* is the large dodecahedron. However, we cannot rule out either of the other two possibilities. Thus *c* is one of the small dodecahedra, but we cannot tell which. Either of these cases is consistent with all our given information. Thus, in the end, we are left with two consistent cases, those depicted in Figure 6.

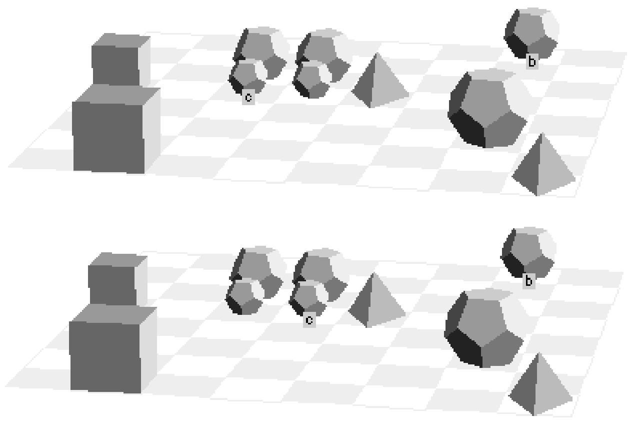


Figure 6: Two remaining possibilities.

## Heterogeneous reasoning

From a pedagogical point of view, exercises like this were quite effective, both in helping the students learn the first-order language (in our class, the English sentences used above would have been replaced by first-order sentences) and in developing some important reasoning skills. The students found the problems challenging but enjoyable, contrary to our experience with the sorts of proofs familiar from elementary logic courses. More important, we discovered that the exercises forced students to focus on the content of the reasoning tasks, rather than the syntactic form of the representations employed.

While this was all very positive, there was a down side. The reasoning students used in solving these problems was of a very different kind from that modeled in standard deductive systems. While it was quite clear when the students' reasoning was correct and when it was faulty, in neither case did it fit the patterns we had come to expect. In other words, the theory of reasoning we were preparing to teach our students seemed inadequate to account for the reasoning the students were already doing to solve the homework problems we set them with Tarski's World.

We explored various ways to treat this reasoning using traditional logic, the logic of first-order sentences. For example, we explored the possibility of considering the visual information presented by the diagram as an efficient way of encapsulating a very large set of sentences. There are two immediate problems with this idea. First, even when it is possible, there are many arbitrary choices to be made in going from a visual representation to a roughly equivalent set of sentences. What language do you use, with which predicates and relations? How do you represent the conclusions, which seem to have the indexical form "*that* block is *d*"? And from the infinite set of sentences true in the diagram, exactly which ones do you choose? In each case, the wrong decision would make the problem unsolvable, and so practically speaking, the conversion to traditional logic could only be accomplished *after* the problem had been solved. This process clearly bore no relation to what our students were doing, nor did it offer a practical tool that would aid them in the future.

The more interesting difficulty concerned the methods of reasoning encountered in this context. If you examine the steps in the above reasoning, it becomes evident that the methods are quite different from those we usually teach in logic. They have a distinctly “semantic” character, quite unlike the syntactically defined methods in formal systems of deduction. For example, you are as likely to break into cases based on atomic, negated or quantified sentences as you are on the basis of a disjunctive claim. As a consequence, attempts to represent the reasoning using sentences do not faithfully model the real character of the reasoning. Simple proofs explode into proofs involving hundreds of steps, proofs in which the key steps in the original reasoning are obscured or obliterated.

We did not abandon our attempt to fit this reasoning into the first-order formalism lightly. As they say, when all you have is a hammer, every problem looks like a nail. But the more we tried reducing the reasoning to sentential reasoning, reasoning amenable to traditional deductive systems, the more we became convinced that the attempted reduction was wrong-headed. Even when we could write down a set of sentences that arguably captured the key features of the blocks world diagram required in the students' reasoning, we realized that those sentences were in fact merely consequences of the diagram, and consequently that the inference from the diagram to the sentences was itself a matter of logic, could itself be valid or invalid. Thus there was a clear sense in which the reasoning our students found so natural was irreducibly *heterogeneous*, involving the interaction of two forms of representation: the Tarski's World diagrams and the first-order sentences. Using traditional sentential systems of deduction, we could at best model *parts* of that reasoning, and even that model did not seem faithful to the actual train of reasoning we witnessed in these exercises.

Looking back at our experience with Turing's World in light of our new-found respect for visualization, we realized that the phenomenon had been staring us in the face there as well. In that case every diagram did have a clear “sentential” counterpart (its 4-tuple representation) but the reasoning involved in constructing or verifying properties of a Turing machine was quite different with the two representational forms. For example, the diagrams make explicit any loops in the program, loops that are not evident in the 4-tuple representation. These loops correspond to recursion and call for proofs by induction. Simply looking at the state diagram leads you to expect when and where induction will be needed to verify a property of the represented machine. Looking at the 4-tuples gives you no such clue.

Once we were sensitized to the power of visualization in reasoning, we realized that it is far more pervasive than we had ever imagined. In everyday life we get information from a variety of sources and in a variety of forms. An important component of everyday reasoning consists in combining these variously presented forms of information. Imagine going to the visitor information center in a city you have never visited and asking for directions. To find your way, you will need to combine the linguistic information from the informant with the visual information gathered from the scenes that unfold before you. Every scientific and engineering discipline has its own system or systems for visualizing information in problem solving, from chemistry's molecular diagrams to geometrical diagrams to ordinary maps and blueprints.

We were reminded, too, that over the years a handful of logicians, most notably Euler, Venn, and Peirce, had stressed the importance and interest of nonsentential inference. The diagrams of Euler and Venn, both of which use circles to represent collections of objects, are still widely known and used, even though their expressive power is sorely limited. C. S. Peirce, inspired by the utility of molecular diagrams in reasoning about chemical compounds, developed a more intricate and powerful diagrammatic formalism. While Peirce's system has not won over many human users, it has become an important tool in computer science.

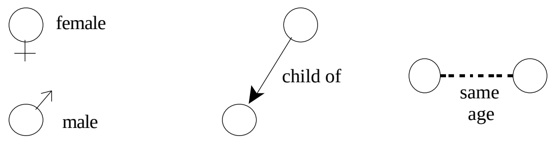
We also looked at the so-called "analytical reasoning" problems posed on standardized tests like the Graduate Record Examination (GRE) and the Law School Aptitude Test (LSAT). These problems are logical puzzles, but the natural way to attack them is almost always to find a good way to represent the information diagrammatically and to use the diagram in reasoning through to a solution. Casting the problem into standard propositional or first-order notation typically obscures the situation, making the solution even harder to find.

## An example

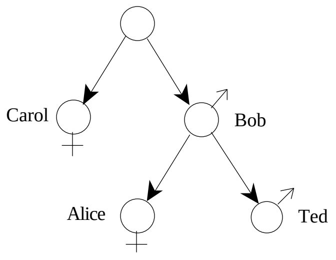
Let's look at an example in some detail. The following analytical reasoning problem is adapted from Summers (1968).

> Bob, his sister Carol, son Ted, and daughter Alice, are all chess players. The best player's twin and the worst player are of the opposite sex. The best player and the worst player are the same age. Can you determine who is best and who is worst?

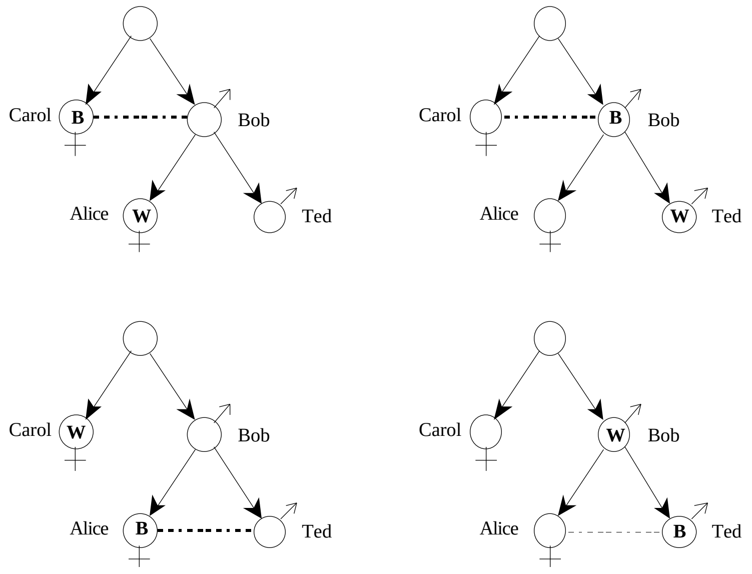
If one approaches this puzzle as a problem in first-order logic, its solution is quite difficult. However, if you use everyday representational devices, the solution is easy to find. We use symbols according to the following conventions:



Using this system of representations, we apply the first sentence of the problem to infer the following diagram, depicting the basic family relationships:



There are various ways to proceed at this point. The most straightforward is to break into four cases, depending on which of the four people is the best chess player. Indicating the best player with a **B** and the worst with a **W**, we end up with the following four diagrams.



Three of these turn out to be inconsistent with the information that the best player and the worst player are the same age, since a father cannot be the same age as his children. The only one that is possible is the third, in which Carol is the worst and Alice is the best.

## 3 What to do?

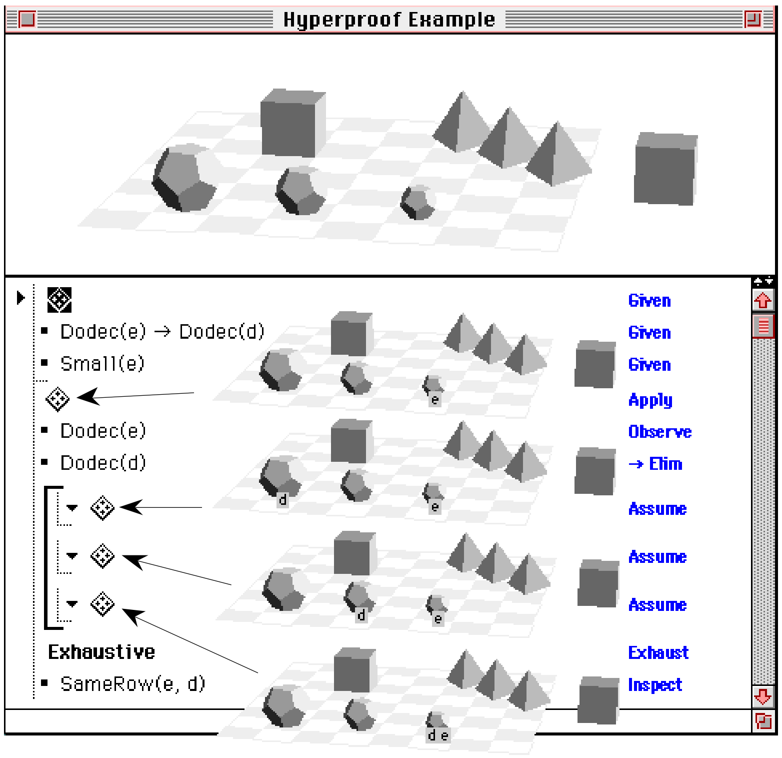
By 1988 we had collected and studied many examples of valid reasoning that did not fit within the confines of logic as it is normally understood. This was both exciting and unsettling. Like most logicians, we thought we understood the basic methods of reasoning, methods like conditional proof, proof by contradiction, proof by cases, and universal generalization. But here were methods of reasoning used every day, methods every bit as basic and important, that did not fit into the conceptual framework with which we were working. Whereas the traditional methods are grouped around the so-called “logical operators,” these new methods seemed to have nothing to do with them. Some of the methods had to do with taking information in sentential form and applying it to modify a diagram, or observing information in a diagram and expressing it with a sentence. Others involved breaking into a range of cases, not on the basis of a disjunctive sentence, but rather on the basis of other kinds of information.

What to do in the face of this discrepancy between logical theory and empirical observation? Rewrite the theory? That was not possible. All we really had were several examples that did not fit comfortably into the current theory. We did not yet have a framework for thinking about, let alone presenting, a richer theory that would encompass all the forms of valid reasoning we found in the wild. But our case studies showed that there were certain methods of reasoning using diagrammatic information combined with sentences that were widely applicable, methods we called observe, apply, exhaustive cases, and so forth.

We became convinced that these principles were at work in reasoning involving many kinds of diagrams and visualization, that these methods were as important as the familiar sentential methods, and that we should be teaching them to our students. For this reason, we initiated the development of a third courseware project, *Hyperproof*, to make it possible to teach these methods of reasoning in addition to the more traditional, sentential methods. We also wrote a paper [5] making the philosophical case for the legitimacy of diagrammatic representations in rigorous reasoning, and a more technical paper [6] sketching the beginnings of a theoretical framework for understanding such reasoning.

## Hyperproof

In Figure 7, we show a simple proof constructed by a student using Hyperproof. This is a proof in which we are given some initial information and asked to determine whether it follows that blocks *d* and *e* are in the same row. As it turns out, it does, and the proof shows this to be the case.



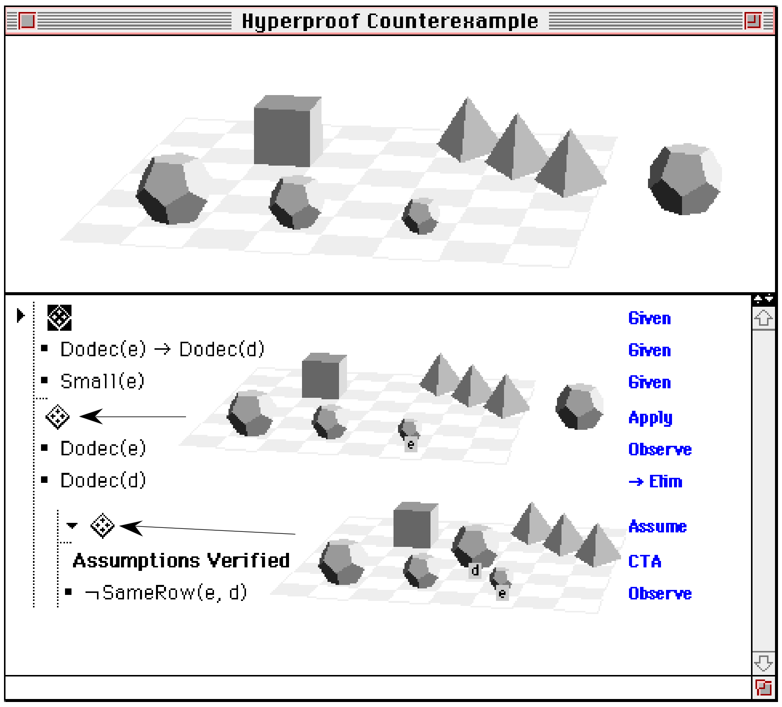
```
                              Given
▪ Dodec(e) → Dodec(d)         Given
▪ Small(e)                    Given
                              Apply
▪ Dodec(e)                    Observe
▪ Dodec(d)                    → Elim
                              Assume
                              Assume
                              Assume
Exhaustive                    Exhaust
▪ SameRow(e, d)               Inspect
```

Figure 7: A proof in Hyperproof.

The initial information is given in the diagram at the top of the figure, plus the two sentences marked as **Given**. The diagram is self-explanatory, except for the meaning of the block depicted to the right of the chess board. This is Hyperproof's way of indicating that this block's location is unknown. The block depicted has a location on the chess board, but the diagram does not tell us where. (Hyperproof has devices for indicating partial information about size and shape, as well.)

The two given sentences assert, first, that *d* is a dodecahedron if *e* is, and second, that *e* is small. The first step in the proof applies this second piece of information to identify *e* as the one small block in the depicted situation. The student then observes that *e* is indeed a dodecahedron. This allows the student to conclude that *d* is a dodecahedron as well. Since there are three dodecahedra, there are three possible cases the student must consider. But in each of these, one can observe that *d* and *e* happen to be in the same row. Hence the conclusion follows.

If we change the initial diagram slightly, then the conclusion would no longer follow. In Figure 8 we have changed the unlocated block to a dodecahedron. In this proof, the student has started in the same way as before, but has discovered that the unlocated block might be *d*. This provides a possible counterexample to the claim that *d* and *e* are in the same row. By placing this block in a different row, the student constructs a possible case in which the premises are true but the purported conclusion is false.



| | |
|---|---|
| | Given |
| Dodec(e) → Dodec(d) | Given |
| Small(e) | Given |
| | Apply |
| Dodec(e) | Observe |
| Dodec(d) | → Elim |
| | Assume |
| Assumptions Verified | CTA |
| ¬SameRow(e, d) | Observe |

Figure 8: A proof of nonconsequence.

An interesting feature of Hyperproof is that it naturally gives rise to many different types of reasoning problems, types that are reflected in the 27 distinct kinds of goals that can be presented with a problem. In addition to the usual problem of the form “show that such-and-such follows from so-and-so,” we can ask “does such-and-such follow from so-and-so?” or “show that such-and-such does not follow from so-and-so.” More interestingly, we can have goals that are expressible only using the diagram, like “can you determine the size of the specified block?” or “can you identify the specified block?” These examples, special as they are to Hyperproof, showed us just how limiting the traditional notion of proof is when it comes to real-world reasoning and problem solving. Of the 27 types of goals available in Hyperproof, only one (the first mentioned above) fits naturally into standard treatments of logic.

## 4 Toward a theory of reasoning

As we said earlier, a theory of inference rich enough to encompass the use of both diagrams and language does not yet exist, but its shape is beginning to emerge. In this final section we want to give our best guess as to the broad shape of this theory and the place the traditional theory will take within the enriched theory.

## Reasoning, proofs, and problem-solving

Two of the key concepts of modern logic are the notions of proof and counterexample (in the form of a model or structure). A proof is used to demonstrate that some piece of information follows from the given information; a counterexample is used to demonstrate that it does not. Notice, however, that these notions do not model reasoning itself, but only two of its possible outcomes.

To see what is left out, think of the legal process. Proofs are the stock-in-trade of the prosecuting attorney, whose aim is to demonstrate that the defendant is guilty. Counterexamples are a standard tool of the defense attorney, whose job is to show that the defendant's guilt has not been established by the evidence. What is left out is the prior role of the detective, who tries to use the available evidence to figure out who is guilty and who is not.

When Sherlock Holmes tries to solve a murder case, he does not start out trying to prove that the butler did it. Rather, his goal is to discover the identity of the murderer, no matter who he or she might be. There is a space of possible suspects and the evidence available to Holmes will either rule out all but one of the suspects, or else be insufficient to determine the murderer. Of course it is not in general a simple matter to figure out which is the case, which is why Holmes is famous for his deductive abilities.

It is often said that modern logic was designed with mathematical reasoning as its paradigm. We think this is not entirely accurate, since the theory provides no better an account of the mathematician's reasoning than it does of everyday reasoning and problem solving. Rather, it is based on the paradigm of mathematical communication: the rigorous proofs (or disproofs) by which one mathematician communicates results to another. This is why the theory, as important as it may be, yields neither a practical tool nor an accurate model of the real-life process of reasoning.

A more accurate and useful theory of reasoning will provide deductively correct methods that can be applied without antecedent knowledge of the ultimate outcome of the reasoning. In other words, the methods should further the goals of the reasoning whether or not they end up producing a proof or a disproof, and whether or not you know ahead of time the specific claim you may ultimately derive. To a limited extent, semantic tableaux have this characteristic: you can apply these methods to a selected sentence to see whether or not it is a consequence of some others. But you still need to choose a specific claim—say, "the butler did it"—before you can apply the method. The deductive system built into Hyperproof shares this advantage of the tableaux method, but also allows you to reason to an initially unspecified conclusion and to do so using information expressed in nonsentential form.

## Exploration, information, and possibilities

When thinking about reasoning and problem solving, a useful metaphor is that of exploration: to solve a reasoning problem, we explore a space of possible situations or worlds consistent with the initial information we are given. Exploration in this context is the attempt to discover information *about* this space of possibilities, information about what it contains, where the boundaries lie, and so forth.

Sentences, whether of English or of first-order logic, partition this space of possibilities, dividing it up into fiefdoms with a multitude of overlapping claims. The study of traditional logic deals with the relations among these claims. Thus, traditional proof techniques allow us to add to a collection of sentences that characterize a set of possibilities, additional sentences that also hold in that set. In this way we can show that the latter information, the information carried by the new sentences, was implicit in the former. The discovery of a counterexample, in contrast, exhibits possible situations consistent with the given information but which falsify the hypothesis in question. In this way we show that the hypothesis is *not* a consequence of the given.

When we view reasoning in this way, it is clear where diagrams and visualization fit in. Diagrams, like sentences, carry information: they carve up the same space of possibilities, though perhaps in very different ways. A good diagram, for example, may represent information in a form that is particularly appropriate for the subject matter at hand, one that allows you to visualize and manipulate the information more readily than would a collection of sentences or even a different sort of diagram. Diagrams useful for depicting relations among sets, say Euler circles, are very different from those used for depicting the structure of a building to be built. The one takes advantage of the inclusion relation among sets; the other takes advantage of spatial relations among parts of a building.

By focusing on sentences, to the exclusion of all other forms of representation, we have neglected one of the most striking facts about the process of reasoning: the heterogeneity of ways in which people represent information in this process. Maps, charts, diagrams, and other nonsentential forms of representation can be, and often are, of equal importance to sentences. Reasoning typically involves the manipulation of information represented both in sentences and various kinds of diagrams. The diagrams play a crucial and legitimate role in both the way the information is presented and in the reasoning itself. This fact has been appreciated in computer science—indeed, database theory is largely the study of the logic of various forms of representation that do not fit neatly into the sentential paradigm. But there is more to be understood about the logic of alternate forms of representation, and it is the natural business of logic to undertake this task.

## Efficiency, informativeness, and complexity

We try to teach our students to be efficient reasoners. But just what does this mean? When reasoning results in a proof, there is an inverse relationship between the efficiency and the complexity of the proof—the latter being measured by some combination of length and maximal depth of nested subproofs. The less complex a proof, the more efficient it is. On the other hand, when reasoning results in a counterexample, there is no ready measure of its complexity, except perhaps size. But the size of a counterexample is not a good measure of the efficiency with which it was discovered.

Our metaphor of reasoning as exploration of a space of possibilities gives another potential metric for efficiency. In this context, more information corresponds to fewer possibilities. To put it the other way around, the more possibilities eliminated by a given proof, or by a given step within a proof, the more information one has extracted, and so the more efficient one is being. Consider, for example, the game of guessing a number between 1 and 100, where you are only allowed to ask "yes/no" questions. A very inefficient method is to ask "Is the number 1?" "Is the number 2?" and so on. The most efficient method is to ask questions like "Is the number greater than 50?" and so on. While the former method eliminates only one number at a time, the latter cuts the possibilities in half. On average, this second method will arrive at the solution faster, though of course users of the first sometimes get lucky.

The notion of informativeness implicit in this strategy is at the basis of Shannon-Weaver communication theory, in which the amount of information in a signal is measured by the number of possibilities eliminated by the signal. The same idea can be used to guide reasoning strategies. In general we can expect that the more informative a given step in a piece of reasoning is, the fewer steps we will need to get to our desired conclusion. This is not an infallible rule, but does give us a good rule of thumb: given a choice of inference steps which are otherwise similar, choose a more informative over a less informative step. This is similar to Grice's conversational maxim of informativeness.

This simple idea has a number of striking applications in everyday reasoning. In our book [4] we include several sections on reasoning strategies in which we use this idea to describe strategies that the students find helpful in their own problem solving. One strategy, for example, helps guide decisions about breaking into diagrammatic cases. By using a maximally informative sentence, you can break into the minimal number of cases, and this, on average, increases the efficiency of your reasoning.

## 5 Conclusion

The natural domain of logic is the study of valid forms of reasoning, methods of extracting new information from information already obtained. Since its inception in Aristotle's *Prior Analytics*, this study has been dominated by a small number of logical systems that apply to information expressed in specific linguistic forms. On the face of it, none of these systems—whether Aristotle's syllogistic, Boole's propositional logic, or the quantified logic of Frege, Peano and Peirce—comes close to accounting for the incredible variety of valid reasoning observed in everyday life.

The history of logic has been a history of squeezing recalcitrant reasoning into existing, well-understood forms. Witness the fact that the paradigmatic syllogism:

All men are mortal.
Socrates is a man.
So Socrates is mortal.

does not itself fit naturally into the theory of syllogisms, since the last two sentences are not of official, Aristotelian form. Yet this did not give two thousand years of Aristotelians pause, because they could easily recast the argument into regulation form:

All men are mortal.
*All things that are Socrates are men.*
So *all things that are Socrates are mortal.*

It is interesting to speculate what would have happened had they questioned this simple conversion, had they recognized the move from the natural form to the official form as involving a logically valid inference, albeit one not accounted for by Aristotelian logic. It is entirely possible that the great advances in quantificational logic of the 20th century might well have occurred centuries before.

Recasting recalcitrant reasoning into well-understood forms can succeed so long as the recalcitrant forms are of less prominence than those that fit the prevailing paradigms, and so long as an alternative account of equal persuasiveness is unavailable. Until recently, reasoning using nonsentential representations could easily be swept under the logician's rug. But the computer revolution has changed this in two main ways. First, the wide variety of representational forms used by computers, both internally and externally, requires us to confront a much richer array of representations, and with them, new forms of valid inference. Consider for example the reasoning involved in extracting sentential information from a city map, a type of reasoning we all engage in from time to time. Prior to the computer, logicians could maintain the fiction that this type of reasoning was, at some deep level, accounted for by first-order logic. But the practical problem of implementing systems to automate this process quickly forces us to give up this fiction.

Second, computers, with their sophisticated graphical capabilities, provide us with powerful tools for constructing, displaying, and even understanding a wide variety of nonsentential representations. In particular, graphical representations become relatively easy to produce and modify, so that a dynamic inference process can be captured and reproduced. It is not an accident that the Hyperproof system was developed on a computer: the computer's graphical capabilities are what makes it practical to create deductive systems employing complex and sophisticated diagrams.

The proper domain of logic is the study of valid forms of information extraction, no matter how that information is represented. Traditionally, logicians have focused on an important, but narrow slice of this domain. In the long run, logic must come to grips with how people use a multitude of representations in rigorous ways. This will force us to extend and enrich the traditional notions of syntax, semantics, logical consequence and proof, in ways that admit these new forms of representation. In the process, what seemed like a finished success story in philosophical and mathematical analysis will be refashioned in exciting new ways.

## Annotated bibliography

1. Allwein, Gerard and Jon Barwise, eds., *Logical Reasoning and Diagrams,* New York: Oxford University Press, 1996.

   A collection of ten recent articles on logical aspects of reasoning using diagrams.

2. Barwise, Jon, and John Etchemendy, *Turing's World*, Stanford: CSLI, and Cambridge: Cambridge University Press, 1993.

   This is a book and computer program for designing Turing machines. The program is described in the article.

3. Barwise, Jon, and John Etchemendy, *Tarski's World,* Stanford: CSLI, and Cambridge: Cambridge University Press, 1991.

This is a book and computer program to help students learn the language of first-order logic. The program is described in the article.

4. Barwise, Jon, and John Etchemendy, *Hyperproof,* Stanford: CSLI, and Cambridge: Cambridge University Press, 1994.

This is a book and computer program to help students learn to reason using both diagrams and the language of first-order logic. The program is described in the article.

5. Barwise, Jon, and John Etchemendy, “Visual Information and Valid Reasoning,” in [17], pp. 9–24. Reprinted in [9], pp. 160–182 and in [1], pp. 3–26.

In this article, the authors argue that diagrams and other visual forms of representations can be legitimate constituents of proofs.

6. Barwise, Jon, and John Etchemendy, “Information, Infons and Inference,” in *Situation Theory and its Applications I,* edited by Robin Cooper and John Perry, Stanford: CSLI Lecture Notes, no. 26 (1989) 33–78.

This article attempts to provide a mathematical underpinning to the argument in [5]. Notions of information and information flow graph are developed and used to give an analysis of some diagrammatic proofs.

7. Barwise, Jon, and John Etchemendy, “Heterogeneous Logic,” in [11], pp. 211–234. Reprinted in [1], pp. 179–200.

This article points out that good diagrammatic representations always exploit features of the domain being represented, and so typically lack the representational expressiveness of language. Rather than being a limitation, this turns out to be an advantage. We give a semantics for Hyperproof, a heterogeneous system, and show why such systems do not need an underlying “interlingua.”

8. Barwise, Jon and Jerry Seligman, *Information Flow: The logic of distributed systems,* New York: Cambridge University Press, 1997.

This book develops a mathematical theory of information flow of the kind that seems to be needed for a theory of inference and proof.

9. Burkholder, Leslie, ed., *Philosophy and the Computer*, Boulder: Westview Press, 1992.

> This volume is a collection of previously published papers which, taken together, show the impact computers were having on philosophy as of 1992. It contains a reprinted version of [5].

10. Etchemendy, John, *The Concept of Logical Consequence*, Cambridge, MA: Harvard University Press, 1990.

> This book argues that the standard analysis of logical consequence is incorrect and that it has severely limited our understanding of the scope of logic.

11. Glasgow, J.I., N.H. Narayanan and B. Chandrasekaran, eds., *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, Menlo Park: AAAI Press, and Cambridge, MA: MIT Press, 1995.

> A collection of twenty-three "recent investigations into the logical, and especially computational, characteristics of diagrammatic representations and the reasoning that can be done with them." The collection grew out of several A.I. workshops on the topic.

12. Hammer, Eric, *Logic and Visual Information*, in Studies in Logic, Language, and Computation, Stanford: CSLI and FoLLI, 1995.

> Hammer presents logical analyses of several existing diagrammatic frameworks, including Venn and Euler circles, Harel's Higraphs, and Peirce's alpha system of diagrams.

13. Leong, Mun-Kew, *Towards a Semantics for a Visual Information System*, Stanford University Ph.D. dissertation, 1994.

> Leong develops a syntax and semantics for a subclass of maps and map-like representations, and develops a preliminary system for visual information retrieval from these representations.

14. Shin, Sun-Joo, *The Logical Status of Diagrams*, Cambridge: Cambridge University Press, 1994.

> Shin's book represents a case study showing that Venn diagrams can be treated with the same rigor and attention to semantic detail as more traditional, sentence-based logics. In this way it provides a rebuttal to the contention that diagrams can be at best a heuristic aid to valid reasoning, not an integral part of it.

15. Shimojima, Atsushi, *On the Efficacy of Representation*, Indiana University Ph.D. dissertation, 1996.

Shimojima explores the relationship between particular representational systems and the domains they represent, pinpointing properties of this relationship that give rise to various inferential properties of these systems. An early version of part of this work appears as a chapter in [1].

16. Summers, George, *New Puzzles in Logical Deduction*, New York: Dover, 1968.

A nice collection of logical puzzles, with solutions, many of them using diagrams.

17. Zimmerman, W. and S. Cunningham, eds., *Visualization in Teaching and Learning Mathematics*, Washington, D.C.: Mathematical Association of America, 1990.

This collection of essays explores the uses of diagrams and other forms of visualization in mathematics education.