Artificial Intelligence 47 (1991) 31–56
Elsevier

Logic and artificial intelligence

Nils J. Nilsson

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Received February 1989

Abstract

Nilsson, N.J., Logic and artificial intelligence, Artificial Intelligence 47 (1990) 31–56.

The theoretical foundations of the logical approach to artificial intelligence are presented. Logical languages are widely used for expressing the declarative knowledge needed in artificial intelligence systems. Symbolic logic also provides a clear semantics for knowledge representation languages and a methodology for analyzing and comparing deductive inference techniques. Several observations gained from experience with the approach are discussed. Finally, we confront some challenging problems for artificial intelligence and describe what is being done in an attempt to solve them.

1. Introduction

Until a technological endeavor achieves a substantial number of its goals, several competing approaches are likely to be pursued. So it is with artificial intelligence (AI). AI researchers have programmed a number of demonstration systems that exhibit a fair degree of intelligence in limited domains (and some systems that even have commercial value). However, we are still far from achieving the versatile cognitive skills of humans. And so research continues along a number of paths—each with its ardent proponents. Although successful AI systems of the future will probably draw upon a combination of techniques, it is useful to study the different approaches in their pure forms in order to highlight strengths and weaknesses. Here, I present my view of what constitutes the "logical approach" to AI.

Some of the criticisms of the use of logic in AI stem from confusion about what it is that "logicists" claim for their approach. As we shall see, logicism provides a point of view and principles for constructing languages and procedures used by intelligent machines. It certainly does not promise a ready-made apparatus whose handle needs only to be turned to emit intelligence. Indeed, some researchers who might not count themselves among those following a logical approach can arguably be identified with the logicist position. (See, for example, Smith's review of a paper by Lenat and Feigenbaum [28, 54].) Other,

more naive, criticisms claim that since so much of human thought is "illogical" (creative, intuitive, etc.), machines based on logic will never achieve human-level cognitive abilities. But puns on the word "logic" are irrelevant for evaluating the use of logic in building intelligent machines; making "illogical" machines is no trouble at all!

In describing logic and AI, we first relate the logical approach to three theses about the role of knowledge in intelligent systems. Then we examine the theoretical foundations underlying the logical approach. Next, we consider some important observations gained from experience with the approach. Lastly, we confront some challenging problems for AI and describe what is being done in an attempt to solve them. For a textbook-length treatment of logic and AI see [12].

2. Artificial intelligence and declarative knowledge

The logical approach to AI is based on three theses:

Thesis 1. Intelligent machines will have knowledge of their environments.

Perhaps this statement is noncontroversial. It is probably definitional. Several authors have discussed what it might mean to ascribe knowledge to machines—even to simple machines such as thermostats [33, 48].

Thesis 2. The most versatile intelligent machines will represent much of their knowledge about their environments declaratively.

AI researchers attempt to distinguish between *declarative* and *procedural* knowledge and argue about the merits of each. (See, for example, [16, 60].) Roughly speaking, declarative knowledge is encoded explicitly in the machine in the form of sentences in some language, and procedural knowledge is manifested in programs in the machine. A more precise distinction would have to take into account some notion of *level* of knowledge. For example, a LISP program which is regarded as a program (at one level) is regarded (at a lower level) as a declarative structure that is interpreted by another program. Settling on precise definitions of procedural and declarative knowledge is beyond our scope here. Our thesis simply states that versatile intelligent machines will have (among other things) a place where information about the environment is stored explicitly in the form of sentences. Even though any knowledge that is ascribed to a machine (however represented in the machine) might be given a declarative interpretation by an outside observer, we will not say that the

machine possesses declarative knowledge unless such knowledge is actually represented by explicit sentences in the memory of the machine.

When knowledge is represented as declarative sentences, the sentences are manipulated by reasoning processes when the machine is attempting to use that knowledge. Thus, the component that decides how to *use* declarative knowledge is separate from the knowledge itself. With procedural approaches to knowledge representation, knowledge use is inextricably intertwined with knowledge representation.

The first serious proposal for an intelligent system with declarative knowledge was by John McCarthy [32]. McCarthy noted the versatility of declaratively represented knowledge: it could be used by the machine even for purposes unforeseen by the machine's designer, it could more easily be modified than could knowledge embodied in programs, and it facilitated communication between the machine and other machines and humans. As he wrote later, "Sentences can be true in much wider contexts than specific programs can be useful" [36].

Smolensky [55] listed some similar advantages: "a. *Public access*: [Declarative] knowledge is accessible to many people; b. *Reliability*: Different people (or the same person at different times) can reliably check whether conclusions have been validly reached; c. *Formality*, *bootstrapping*, *universality*: The inferential operations require very little experience with the domain to which the symbols refer."

To exploit these advantages, the declaratively represented knowledge must, to a large extent, be *context free*. That is, the *meaning* of the sentences expressing the knowledge should depend on the sentences themselves and not on the external context in which the machine finds itself. The context-free requirement would rule out terms such as "here" and "now" whose meaning depends on context. Such terms are called *indexicals*.

Many database systems and expert systems can be said to use declarative knowledge, and the "frames" and "semantic networks" used by several AI programs can be regarded as sets of declarative sentences. On the other hand, there are several examples of systems that do not represent knowledge about the world as declarative sentences. Some of these are described in the other papers in this volume.

Thesis 3. For the most versatile machines, the language in which declarative knowledge is represented must be at least as expressive as first-order predicate calculus.

One might hope that a natural language such as English might serve as the language in which to represent knowledge for intelligent systems. If this were possible, then all of the knowledge already compiled in books would be immediately available for use by computers. Although humans somehow

understand English well enough, it is too ambiguous a representational medium for present-day computers—the meanings of English sentences depend too much on the contexts in which they are uttered and understood.

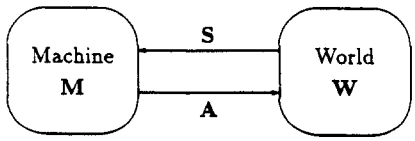
AI researchers have experimented with a wide variety of languages in which to represent sentences. Some of these languages have limited expressive power. They might not have a means for saying that one or another of two facts is true without saying which fact is true. Some cannot say that a fact is not true without saying what is true instead. They might not be able to say that *all* the members of a class have a certain property without explicitly listing each of them. Finally, some are not able to state that at least one member of a class has a certain property without stating which member does. First-order predicate calculus, through its ability to formulate disjunctions, negations, and universally and existentially quantified sentences, does not suffer from these limitations and thus meets our minimal representational requirements.

3. Foundations of the logical approach

In addition to the three theses just stated, the logical approach to AI also embraces a point of view about what knowledge is, what the world is, how a machine interacts with the world, and the role and extent of special procedures in the design of intelligent machines.

Those designers who would claim that their machines possess declarative knowledge about the world are obliged to say something about what that claim means. The fact that a machine's knowledge base has an expression in it like $(\forall x)\mathsf{Box}(x) \supset \mathsf{Green}(x)$, for example, doesn't by itself justify the claim that the machine *believes* all boxes are green. (The mnemonic relation constants that we use in our design aren't mnemonic for the machine! We could just as well have written $(\forall x)\mathsf{GO11}(x) \supset \mathsf{GO23}(x)$.)

There are different views of what it means for a machine possessing a database of sentences to believe the facts intended by those sentences. The view that I favor involves making some (perhaps unusual) metaphysical



$see : \mathcal{W} \longrightarrow \mathcal{S}$

$mem : \mathcal{S} \times \mathcal{M} \longrightarrow \mathcal{M}$

$act : \mathcal{S} \times \mathcal{M} \longrightarrow \mathcal{A}$

$effect : \mathcal{A} \times \mathcal{W} \longrightarrow \mathcal{W}$

Fig. 1. Machine and world.

assumptions about what we take the "real world" to be and about how our machines interact with that world. I will give a simplified account of this view here. It is based, in part, on a discussion of intelligent agent architecture in [12, Chapter 13].

Figure 1 shows a machine interacting with the world. Both machine and world are regarded as finite-state machines. We denote the machine state by M; it is one of a set $\mathcal{M}$ of states. We denote the world state by W; it is one of a set $\mathcal{W}$ of states. The input to the machine is denoted by S—one of a set $\mathcal{S}$ of inputs; the output of the machine is denoted by A—one of a set $\mathcal{A}$ of outputs. States, inputs, and outputs are related as follows: the function *see* maps $\mathcal{W}$ into $\mathcal{S}$; the function *mem* maps $\mathcal{S} \times \mathcal{M}$ into $\mathcal{M}$; the function *act* maps $\mathcal{S} \times \mathcal{M}$ into $\mathcal{A}$; lastly, the function *effect* maps $\mathcal{A} \times \mathcal{W}$ into $\mathcal{W}$. The function *see* models the fact that the machine is not sensitive to every aspect of the world; it partitions the world into classes whose members, as far as the machine is concerned, are equivalent. The function *mem* models the machine's memory behavior; the machine's state at any instant is a function of the machine's input and its previous state. The function *act* describes how the machine acts on the world; its action is a function of its state and its input. We model the effects of these actions on the world (as well as the world's own internal dynamics) by *effect*.

This model of a machine and its world is sufficiently general to capture a number of approaches to intelligent machine design. To particularize it to the logical approach, we stipulate that the state of the machine is given by a set of sentences, which for concreteness we hereinafter take to be sentences in the first-order predicate calculus. The function *mem* transforms such a set of sentences (together with the input to the machine) into another set of sentences (and thus changes the machine's state). The function *act* is a function of such a set of sentences (and the machine's input) and produces as output a machine action.

Describing how the designer specifies *act*, *mem*, and an initial set of sentences requires some discussion about the relationship between these sentences and what the designer imagines the world of the machine to be like. We suppose that the designer thinks of the world literally as a finite-state machine which he describes as a mathematical structure consisting of *objects*, *functions*, and *relations*. Some of the objects in this mathematical structure might be states, others might be other entities that the designer thinks exist in the world—some of which are dependent on state. This structure must also account for the finite-state machine function *effect*, which produces a new world state depending on the action of the intelligent machine and the old world state, and the function *see*, which maps world states into the input to the intelligent machine. AI researchers have explored a variety of ways to conceive of the world in terms of objects, functions, and relations; while we will not describe any particular conceptualization here, they can all be accommodated by the account we are giving. (It may seem strange to think of the *real* world as

a mathematical structure, but since our picture provides for the world to be affected by and affect itself and the intelligent machine, one shouldn't worry that our view of the world is impractically ethereal.)

Now, the designer of a machine that is to interact with the world never knows what the world objects, functions, and relations actually are. He must guess. Guessing involves *invention* on the designer's part. (Our machine designer is in the same predicament as is the scientist; scientists invent descriptions of the world and gradually refine them until they are more useful.) We use the term *conceptualization* to describe the designer's guess about the world objects, functions, and relations. The designer may not even be able to specify a single conceptualization; for example he may choose not to commit himself regarding whether an object he invents, say a block, has the color property green or blue. Thus, in general, the designer attempts to specify a set of conceptualizations such that, whatever the world actually is, he guesses it is a member of the set.

The designer realizes, of course, that his conceptualization might not accurately capture the world—even as he himself believes it to be. For example, his conceptualization may not discriminate between objects that he himself recognizes to be different but which can be considered to be the same considering his purposes for the machine. The designer need only invent a conceptualization that is *good enough*, and when and if it becomes apparent that it is deficient (and that this deficiency is the cause of inadequate machine performance), he can modify his conceptualization.

We stress that the objects guessed to exist in the world by the designer are *invented*. He is perfectly free to invent anything that makes the machine perform appropriately, and he doesn't ask whether or not some object *really* does or does not exist (whatever that might mean) apart from these invented structures. For many ordinary, concrete objects such as chairs, houses, people, and so on, we can be reasonably confident that our inventions mirror reality. But some of the things that we might want to include as world objects, such as *precambrian unconformities*, English sentences, *the Peloponnesian War*, π, and *truth*, have a somewhat more arbitrary ontological status. In fact, much of the designer's guess about the world may be quite arbitrary in the sense that other guesses would have suited his purposes equally well. (Even those researchers following other declarative, but putatively non-logical, approaches must invent the equivalent of objects, relations, and functions when they attempt to give their machines declarative knowledge.)

A logicist expresses his conceptualization of the world (for the machine) by a set of sentences. The sentences are made part of the machine's memory (comprising its state) and embody the machine's declarative knowledge. We assume that the sentences are in the first-order predicate calculus; this language and the sentences in it are constructed as follows: For every world object in the conceptualization we create an *object constant*; for every world relation, we

create a *relation constant*; and for every world function, we create a *function constant*. Using these constructs, and the syntax of predicate calculus, we (the designer) then compose a set of sentences to express the declarative knowledge that we want the machine to have about the world.

When a designer cannot (or does not choose to) specify which of two relations holds, he uses a disjunction, such as: $\mathsf{Box(Ob1)} \wedge [\mathsf{Blue(Ob1)} \vee \mathsf{Green(Ob1)}]$. Or he may use an existentially quantified statement; $(\exists x)\mathsf{Box(x)} \wedge \mathsf{Green(x)}$. Or, he might know that all boxes are green: $(\forall x)\mathsf{Box(x)} \supset \mathsf{Green(x)}$.

In what sense can we say that a collection of predicate calculus sentences represents *knowledge* about the world? Our answer to this question involves the notions of *interpretations* and *models* of sentences in the predicate calculus. Briefly, an interpretation consists of:

(1) an assignment of a relation to each relation constant;
(2) an assignment of an object to each object constant;
(3) an assignment of a function to each function constant;
(4) a procedure for assigning the values T (true) or F (false) to each closed formula. (This procedure involves *evaluating* ground atomic formulas using the relation/object/function assignments and then using the standard logical truth tables for non-atomic ground sentences. The description of how quantified sentences are evaluated is slightly more complex but respects the intuitive meanings of the quantifiers.)

Any interpretation for which all of the sentences in a set of sentences evaluates to T is called a *model* of the set of sentences.

In terms of these definitions, the designer's task can be re-stated as follows:

> Invent world objects, relations, and functions; a first-order predicate calculus language; an interpretation of the expressions of this language in terms of the objects, relations, and functions; and then compose a set of sentences in the language such that the interpretation of those sentences is a model of the set of sentences.

We will call the world objects, relations, and functions invented by the designer, the *intended model* of the sentences the designer uses to describe the world. Although this interpretation itself may never actually be represented explicitly as mathematical structure, it is important that it be firmly fixed in the mind of the designer. With this interpretation in mind, the designer invents linguistic terms to denote his invented objects, functions, and relations and writes down predicate calculus sentences for which the intended interpretation is a model.

The designer gives the machine declarative knowledge about the world by storing these sentences in the machine's memory. We call the set of sentences the *knowledge base* of the machine and denote the set by Δ. We assume that the designer fixes the initial state of the machine by specifying some Δ_0; when

the machine is attached to the world, as in Fig. 1, *mem* produces a sequence of states $\Delta_0, \Delta_1, \ldots, \Delta_i, \ldots$.

Even when the designer has a single intended interpretation in mind, Δ, in general, will be satisfied by a set of interpretations—the intended one among them. The designer must provide sufficient sentences in the knowledge base such that its models are limited—limited so that even though the set has more than one model, it doesn't matter given the purposes for the machine. (To the extent that it *does* matter, the designer must then provide more sentences.) In designing knowledge bases, it frequently happens that the designer's idea of the intended interpretation is changed and articulated by the very act of writing down (and reasoning with) the sentences.

So, a machine possessing a set of sentences *knows* about the world in the sense that these sentences admit of a set of models, and this set is the designer's best approximation to what the world actually is, given the purposes for the machine. The *actual* world might not even be in the set (the designer's guess might be wrong), so we really should be talking about the machine's *beliefs* rather than the machine's *knowledge*. But, following the tradition established by the phrase "knowledge-based systems," we will continue to speak of the machine's knowledge.

The machine's procedural knowledge is represented in the functions *mem* and *act*. The function *mem* changes the sentences and thereby changes the machine's state. Perhaps new sentences are added or existing ones are modified or deleted in response to new sensory information. The function *mem* may also produce a change in the machine's state in the absence of sensory information; changes to Δ may occur through processes of deduction or other types of inference as will be described below.

The machine's declarative knowledge affects its actions through the function *act*. We take *act* to be a function (over sets of sentences) that produces actions. Note that *act* can thus only respond to sentences *qua sentences*, that is, as strings of symbols. It is not a function of the models of these sentences!

Given this picture, we can identify a spectrum of design choices. At one end, *act* and *mem* are highly specialized to the tasks the machine is expected to perform and to the environment in which it operates. We might say, in this case, that the machine's knowledge is mainly *procedurally* represented. At the other extreme, *act* and *mem* are general purpose and largely independent of the application. All application-specific knowledge is represented in Δ. The machine's knowledge in this case can be said to be mainly *declaratively* represented. The logical approach usually involves a commitment to represent most of the machine's knowledge declaratively. For a proposal at the extreme declarative end, see [12, Chapter 13]. It is not yet known to what extent this goal can be achieved while maintaining reasonable efficiency.

Because the actions emitted by *act* depend on the syntactic form of the sentences in Δ, it is necessary for *mem* to be able to rewrite these sentences in

the form appropriate to the task at hand. This aspect of *mem* we call *reasoning*. Imagine, for example, a robot designed to paint boxes green. Its sentence-to-action process, *act*, may include a *production rule* like "If Δ includes the sentence Box(η) for some value of η, paint the object denoted by η green." But suppose Δ includes the sentences (∀x)Blue(x) ⊃ Box(x) and Blue(G17) but not Box(G17) explicitly. We might expect that correct behavior for this robot would be to paint the object denoted by G17 green, but there is no sentence-to-action rule to accomplish that unless Box(G17) occurs explicitly in Δ. Constructing the sentence Box(G17) from the sentences (∀x)Blue(x) ⊃ Box (x) and Blue(G17) is an example of one kind of sentence manipulation, or *inference*, that we want *mem* to do.

Often, as in the box-painting example, the new sentence constructed from ones already in memory does not tell us anything new about the world. (All of the models of (∀x)Blue(x) ⊃ Box(x) and Blue(G17) are also models of Box(G17). Thus, adding Box(G17) to Δ does not reduce the set of models.) What the new sentence tells us was already implicitly said by the sentences from which it was constructed.

If all of the models of Δ are also models of a sentence ϕ, we say that Δ *logically entails* ϕ and write $\Delta \models \phi$. Among the computations that we might want *mem* to perform are those which add sentences to Δ that are logically entailed by Δ. One apparent problem in devising such computations is the prospect of having to check *all* the models of Δ to see if they are also models of ϕ. But, fortunately, there exist strictly syntactic operations on Δ that are able to compute logically entailed formulas.

We use the phrase *rule of inference* to refer to any computation on a set of sentences that produces new sentences. If ψ can be derived from Δ by a sequence of applications of rules of inference, we say that ψ can be *deduced* from Δ and write $\Delta \vdash \psi$. An example is the rule of inference called *modus ponens*. From any sentences of the form $\rho \supset \sigma$ and ρ, we can deduce the sentence σ by modus ponens. The process of logical deduction involves using a set of rules of inference to deduce additional sentences from a set of sentences. Interestingly, it happens that there are rules of inference, modus ponens is an example, that have the property that if $\Delta \vdash \phi$, then $\Delta \models \phi$. Such rules of inference are called *sound*.

Sound rules of inference are extremely important because they allow us to compute sentences that are logically entailed by a set of sentences using computations on the sentences themselves (and not on their models).

We can also find sets of inference rules that have the property that if $\Delta \models \phi$ then the rules (successively applied) will eventually produce such a ϕ. Such a set of inference rules is called *complete*.

Although all logicists typically incorporate sound inference rules as part of the calculations performed by *mem*, there is no necessary reason to limit *mem* to performing sound inferences. Other computations are often desirable. We will describe some of these later in the paper.

In summary, intelligent machines designed according to the logical approach are state-machines whose states are sets of sentences. Machine state transitions are governed by a function, *mem*, acting on the sentence sets and the inputs to the machine. An important, but not the only, component of *mem* is sound logical inference. Machine actions are governed by a function, *act*, of the machine's state and inputs. The intended interpretation of the sentences in a machine's state involves objects, functions, and relations that are the designer's guesses about the world.

> Through naming comes knowing; we grasp an object, mentally, by giving it a name—hension, prehension, apprehension. And thus through language create a whole world, corresponding to the other world out there. Or we trust that it corresponds. Or perhaps, like a German poet, we cease to care, becoming more concerned with the naming than with the things named; the former becomes more real than the latter. And so in the end the world is lost again. No, the world remains—those unique, particular, incorrigibly individual junipers and sandstone monoliths—and it is we who are lost. Again. Round and round, through the endless labyrinth of thought—the maze. (Edward Abbey [1, pp. 288–289].)

4. Comments on the logical approach

The basic idea underlying the logical approach to AI is simple, but attempts to use it have resulted in several additional important insights.

4.1. The importance of conceptualization

The most important part of "the AI problem" involves inventing an appropriate conceptualization (intended model). It is not easy for a designer to squeeze his intuitive and commonsense ideas about the world into a coherent conceptualization involving objects, functions, and relations. Although this exercise has been carried out for several limited problem domains (most notably those to which expert systems have been successfully applied), there are some particularly difficult subjects to conceptualize. Among these are liquids and other "mass substances," processes, events, actions, beliefs, time, goals, intentions, and plans. Some researchers feel that the *frame problem*, for example, arises as it does as an artifact of an inappropriate (state-based) conceptualization of change [17]. Others feel that change must involve the notion of time (instead of the notion of state) [52]. Conceptualizing the "cognitive state" of intelligent agents has been the subject of recent intense study. (See, for example, [8] for a treatment of the intentions of agents and [24, 40] for treatments of the knowledge and beliefs of agents.) Interestingly,

many of the most difficult conceptualization problems arise when attempting to express knowledge about the everyday, "commonsense" world (see [20, 21]). AI researchers join company with philosophers who have also been attempting to formalize some of these ideas.

Choosing to use first-order predicate calculus as a representation language does not relieve us of the chore of deciding *what* to say in that language. Deciding what to say is harder than designing the language in which to say it! The logical approach to AI carries with it no special insights into what conceptualizations to use. (Logic is often criticized for providing *form* but not *content*. Of course!)

It is important to stress that these conceptualization problems do not arise simply as an undesirable side effect of the use of logic. They must be confronted and resolved by any approach that attempts to represent knowledge of the world by sentence-like, declarative structures. The fact that these problems are exposed quite clearly in the coherent framework provided by the logical approach should be counted as an advantage.

4.2. Sound and unsound inferences

Another important observation concerns the subject of sound inference. Logicists are sometimes criticized for their alleged dependence on deduction. Much human thought, the critics rightly claim, involves leaps of intuition, inductive inference, and other guessing strategies that lie outside the realm of sound inference. There are two things that can be said about such criticism.

First, logicists regard sound inference as an important, but not the only, component of reasoning. We must be careful to note the circumstances under which both sound and unsound inferences might appropriately be used. Recall that the set of sentences Δ (with which a designer endows a machine) implicitly defines a set of models. Either the designer actually has some subset of these models in mind (as his guess about what the world is) or he is completely unbiased about which of the models might represent the world. If he really is unbiased, nothing other than sound inference would be desired by the designer. Any deduced sentence ϕ had better be logically entailed by Δ; if there are some models of Δ, for example, that are not models of ϕ, and if the designer wanted the machine to conclude ϕ, then he wouldn't have been completely unbiased about which of the models of Δ represented the world.

If the designer has some subset of the models of Δ in mind, and if (for one reason or another) he could not specify this subset by enlarging Δ, then there are circumstances under which unsound inference might be appropriate. For example, the designer may have some preference order over the models of Δ. He may want to focus, for example, on the minimal models (according to the preference order). These minimal models may be better guesses, in the designer's mind, about the real world than would be the other models of Δ. In

that case, the inference ϕ would be appropriate if all the minimal models of Δ were models of ϕ. (We will see an example of the use of minimal models later.)

McCarthy [34] and Lifschitz [29] have studied a variety of such preference orders over models and have investigated an (unsound) inference computation, called *circumscription*, that is based on them. (We describe circumscription later in this paper.) Several aspects of commonsense reasoning, including inductive inference and default inference, appear to be mechanizable using circumscription or something very much like it.

Although inductive inference is a complex and well studied subject, we can use a simple version of it as an example of unsound inference. Consider the premises

$$\begin{aligned}&\mathsf{Emerald(Ob1)} \wedge \mathsf{Color(Ob1, Green)},\\ &\mathsf{Emerald(Ob2)} \wedge \mathsf{Color(Ob2, Green)},\\ &\ldots,\\ &\mathsf{Emerald(Obn)} \wedge \mathsf{Color(Obn, Green)}.\end{aligned}$$

For some adequately large value of n, we may want to inductively infer (unsoundly but reasonably) $(\forall x)\mathsf{Emerald}(x) \supset \mathsf{Color}(x, \mathsf{Green})$ if there is no η mentioned in Δ such that Δ entails $\mathsf{Emerald}(\eta) \wedge \neg\mathsf{Color}(\eta, \mathsf{Green})$.

The second thing that can be said in defense of sound inference is that, in the context of sufficient additional information, sound conclusions can be drawn that might have seemed to have required unsound inference without the additional information. For example, suppose Δ contains the following statements:

$$(\exists y)(\forall x)\mathsf{Emerald}(x) \supset \mathsf{Color}(x, y)$$

(intended meaning: there is a color such that all emeralds have that color),

$$(\forall x, y, z)[\mathsf{Color}(x, y) \wedge \mathsf{Color}(x, z)] \supset (y = z)$$

(intended meaning: a thing can have only one color).

From these statements, we can deduce that if a thing is an emerald, it has a unique color. Now, if we subsequently learn

$$\mathsf{Color(Ob1, Green)} \wedge \mathsf{Emerald(Ob1)},$$

we can deduce (soundly) $(\forall x)\mathsf{Emerald}(x) \supset \mathsf{Color}(x, \mathsf{Green})$. Δ already told us that all emeralds have the same unique color, but did not specify what that color was. Later information added to Δ allowed us to deduce a more specific general rule.

Although the logical approach might sanction unsound inferences, logicists would certainly want such inferences to have a well motivated model-theoretic

justification. For example, circumscription is motivated by minimal-model entailment and thus might be called a "principled" inference even though not a sound one.

4.3. Efficiency and semantic attachment to partial models

Earlier, we mentioned that it was fortunate that sound inference techniques existed because it is impossible in most situations to check that all the models of Δ were also models of some formula ϕ. This "good fortune" is somewhat illusory however, because finding deductions is in general intractable and for many practical applications unworkably inefficient. Some people think that the inefficiency of the logical approach disqualifies it from serious consideration as a design strategy for intelligent machines.

There are several things to be said about logic and efficiency. First, it seems incontestable that knowledge can be brought to bear on a problem more efficiently when its use is tailored to the special features of that problem. When knowledge is encoded in a fashion that permits many different uses, several possible ways in which to use it may have to be tried in any given situation, and the resulting search process takes time. A price does have to be paid for generality, and the logical approach, it seems, pays a runtime cost to save accumulated design costs.

But even so, much progress has been made in making inference processes more efficient and practical for large problems. Stickel has developed one of the most powerful first-order-logic theorem provers [56, 57]. Several resolution refutation systems have been written that are able to solve large, nontrivial reasoning problems, including some open problems in mathematics [59, 61]. Many large-scale AI systems depend heavily on predicate calculus representations and reasoning methods. Among the more substantial of these are TEAM, a natural language interface to databases [14]; DART, a program for equipment design and repair [11]; and KAMP, a program that generates English sentences [3].

A very important technique for achieving efficiency in the context of the logical approach involves augmenting theorem-proving methods with calculations on model-like structures. Often, calculations on models are much more efficient than are inference processes, and we would be well advised to include them as part of a machine's reasoning apparatus.

We mentioned that seldom does a designer make explicit his guess about the world, the intended model. The set of models is implicitly defined by the set of sentences in Δ. Sometimes, however, it is possible to be explicit about at least part of the intended model. That is, we might be able to construct a part of the model as list structure and programs in, say, LISP. For example, we can represent objects as LISP atoms, functions as LISP functions, and relations as LISP predicates. In such cases we can perform reasoning by computations with

these LISP constructs that we might otherwise have performed by logical inference on the predicate calculus sentences. This point has previously been made by a number of researchers[1]—most notably by Weyhrauch [58]. Typically, the LISP constructs will constitute only a *partial* model; that is, there might not be LISP referents for the expressions in all of the predicate calculus sentences, and/or the LISP functions and predicates themselves may be defined over just a subset of the intended domain. (In the following we will not always be careful about saying *partial* models and interpretations even though those are what we really mean.)

As an example, consider the predicate calculus sentences:

$$P(A), \qquad (\forall x)P(x) \supset Q(x) .$$

Presumably, these sentences stem from a certain conceptualization of the world. Suppose we can capture this conceptualization in LISP. Our intended interpretation for P is (represented by) a certain LISP predicate ***P***; our intended interpretation for Q is a certain LISP predicate ***Q***; and our intended interpretation for A is a certain LISP atom ***A***. If these intended interpretations are to be parts of models for Δ, then we know that (***P A***) and (**or** (**not** (***P A***)) (***Q A***)) will both evaluate to *T*. Thus, we must make sure that (***Q A***) evaluates to *T*.

So, this gives us two ways to compute the truth value of Q(A) with respect to the intended model. Given the initial sentences, we could deduce Q(A) using sound inference rules. That is, if a reasoning program needed to know whether or not Q(A) was true, it could find that it is true using logical inference. The fact that Q(A) can be (soundly) deduced from the other sentences means that Q(A) is logically entailed by them. And that means that *all* of the models of the initial sentences are also models of Q(A). That means, *a fortiori*, that the *intended* model of the initial sentence (whatever it is) is a model of Q(A).

The other way to compute the truth value of Q(A) is to associate Q(A) with (***Q A***) and evaluate (***Q A***) in LISP. We call this association *semantic attachment*. Semantic attachment to appropriate computational structures, when such are available, is sometimes more efficient than inference. It provides a means for determining the truth of an expression directly in the intended model rather than by establishing its truth in all models indirectly by sound inference.

It is my guess that practical AI systems will use a combination of logical inference and semantic attachment with the latter perhaps predominating in some applications and the former used as a "fall-back" method of great generality. Several standard structures and programs already commonly used in AI systems can be employed in semantic attachments. For example, tree structures are useful for representing taxonomic hierarchies (in fact, some knowledge representation languages use such tree-structure representations

[1] For a comprehensive recent treatment of attachment see: K.L. Myers, Automatically generating universal attachments through compilation, in: *Proceedings AAAI-90*, Boston, MA (1990) 252–257.

explicitly as part of the language [5]). Various LISP ordering predicates combined with appropriate directed-graph data structures are useful for representing transitive binary relations.

4.4. Reification of theories

Sometimes we will want our machines to reason about (rather than with) the sentences in its knowledge base. We may, for example, want them to reason about the lengths of sentences or about their complexity. Our conceptualizations will thus have to acknowledge that things called sentences exist in the world. Conferring *existence* on abstract concepts (such as sentences) is often called *reification*.

We might reify whole theories. This will allow us to say, for example, that some Δ_1 is more appropriate than is some Δ_2 when confronted with problems of diagnosing bacterial infections. Scientists are used to having different—even contradictory—theories to explain reality: quantum physics, Newtonian mechanics, relativity, wave theories of light, particle theories of light, and so on. Each is useful in certain circumstances. Although scientists search for a uniform, all-embracing, and consistent picture of reality, historically they have had to settle for a collection of somewhat different theories. There is nothing in the logicist approach that forces us, as machine designers, to use just *one* conceptualization of the world. There is no reason to think AI would be any more successful at that goal than scientists have been!

When theories are reified, *metatheory* (that is, a theory about theories) can be used to make decisions about which local theory should be used in which circumstances. For example, the metatheory might contain a predicate calculus statement having an intended meaning something like: "When planning a highway route, use the theory that treats roads as edges in a graph (rather than, for example, as solid objects made of asphalt or concrete)". Metatheory can also provide information to guide the inference procedures operating over local theories. For example, we might want to say that when two inferences are possible in some Δ_1, the inference that results in the most general conclusion should be preferred. Using metatheory to express knowledge about how to control inference is consistent with the logicists' desire to put as much knowledge as possible in declarative form (as opposed to "building it in" to the functions *mem* and *act*).

Weyhrauch [58] has pointed out that the process of semantic attachment in a metatheory can be particularly powerful. Commonly, even when no semantic attachments are possible to speed reasoning in a theory, the problem at hand can be dispatched efficiently by appropriate semantic attachment in the metatheory.

Some critics of the logical approach have claimed that since *anything* can be said in the metatheory, its use would seem to be a retreat to the same ad hoc

tricks used by less disciplined AI researchers. But we think there are generally useful things to say in the metatheory that are not themselves problem dependent. That is, we think that knowledge about how to use knowledge can itself be expressed as context-free, declarative sentences. (Lenat's work has uncovered the best examples of generally useful statements about how to use knowledge [25–27].)

4.5. Other observations

Even though they frequently call the sentences in their knowledge bases *axioms*, logicists are not necessarily committed to represent knowledge by a minimal set of sentences. Indeed, some (or even most) of the sentences in Δ may be derivable from others. Since the "intelligence" of an agent depends on how much usable declarative knowledge it has, we agree completely with those who say "In the knowledge lies the power." We do not advocate systems that rely on search-based derivations of knowledge when it is possible to include the needed knowledge explicitly in the knowledge base. The use of very large knowledge bases, of course, presupposes efficient retrieval and indexing techniques.

The occasional criticism that logicists depend too heavily on their inference methods and not on the knowledge base must simply result from a misunderstanding of the goals of the logical approach. As has already been pointed out, logicists strive to make the inference process as uniform and domain independent as possible and to represent all knowledge (even the knowledge about how to use knowledge) declaratively.

5. Challenging problems

5.1. Language and the world

Few would deny that intelligent machines must have some kind of characterization or model of the world they inhabit. We have stressed that the main feature of machines designed using the logical approach is that they describe their worlds by *language*. Is language (any language) adequate to the task? As the writer Edward Abbey observed [1, p. x]:

> Language makes a mighty loose net with which to go fishing for simple facts, when facts are infinite.

A designer's intuitive ideas about the world are often difficult to capture in a conceptualization that can be described by a finite set of sentences. Usually these intuitive ideas are never complete at the time of design anyway, and the conceptualization expands making it difficult for the sentences to catch up.

John McCarthy humorously illustrates this difficulty by imagining how one

might formulate a sentence that says that under certain conditions a car will start. In English we might say, for example: "If the fuel tank is not empty and if you turn the ignition key, the car will start." But this simple sentence is not true of a world in which the carburetor is broken, or in which the fuel tank (while not empty) is full of water, or in which the exhaust pipe has a potato stuck in it, or . . . . Indeed, it seems there might be an infinite number of *qualifications* that would need to be stated in order to make such a sentence true (in the world the designer has in mind—or comes to have in mind). Of course, just what it means for a designer to have a world in mind is problematical; he probably didn't even think of the possibility of the potato in the tailpipe until it was mentioned by someone else who happened to conceive of such a world.

There seem to be two related problems here. One is that we would like to have and use approximate, simple conceptualizations even when our view of the world would permit more accurate and detailed ones. The approximate ones are often sufficient for our purposes. Thus, even though we know full well that the carburetor must be working in order for a car to start, in many situations for which we want to reason about the car starting we don't need to know about the carburetor and can thus leave it out of our conceptualization. Using theories (Δ's) corresponding to approximate conceptualizations and successive refinements of them would seem to require the ability to have several such at hand and a metatheory to decide when to use which.

Another problem is that even the most detailed and accurate conceptualization may need to be revised as new information becomes available. Theories must be revisable to accomodate the designer's changing view of the world. As the machine interacts with its world, it too will learn new information which will in some cases add to its theory and in other cases require it to be modified.

Science has similar problems. Scientists and engineers knowingly and usefully employ approximate theories—such as frictionless models. Furthermore, all of our theories of the physical world are falsifiable, and, indeed, we expect scientific progress to falsify the theories we have and to replace them by others. When we conclude something based on a current physical theory, we admit the dependence of the conclusion on the theory and modify the theory if the conclusion is contradicted by subsequent facts. Those who would argue that logical languages are inappropriate for representing synthetic or contingent knowledge about the world [39] would also seem to have to doubt the utility of any of the languages that science uses to describe and predict reality. Merely because our conceptualization of the world at any stage of our progress toward understanding it may (inevitably will!) prove to be inaccurate does not mean that this conceptualization is not in the meantime useful.

Some AI researchers have suggested techniques for making useful inferences from an approximate, but not inaccurate, theory. We say that a theory is not inaccurate if its models include the world as conceived by the designer. If a

theory is to be not inaccurate, it is typically impossible or overly cumbersome to include the universal statements needed to derive useful sound conclusions.

We illustrate the difficulty by an example. Suppose that we want our machine to decide whether or not an apple is edible. If Δ is to be not inaccurate, we cannot include in it the statement $(\forall x)\mathsf{Apple}(x) \wedge \mathsf{Ripe}(x) \supset \mathsf{Edible}(x)$ in the face of known exceptions such as $\mathsf{Wormy}(x)$ or $\mathsf{Rotten}(x)$. (We trust that the reader understands that the mnemonics we use in our examples must be backed up by sufficient additional statements in Δ to insure that these mnemonics are constrained to have roughly their intended meanings.) Suppose we cannot conclude from Δ that a given apple, say the apple denoted by apple1 is wormy or rotten; then we may want to conclude (even non-soundly) $\mathsf{Edible}(\mathsf{Apple1})$. If later, it is learned (say through sensory inputs) that $\mathsf{Rotten}(\mathsf{Apple1})$, then we must withdraw the earlier conclusion $\mathsf{Edible}(\mathsf{Apple1})$. The original inference is called *defeasible* because it can be defeated by additional information. Making such inferences involves what is usually called *nonmonotonic reasoning*. (Ordinary logical reasoning is monotonic in the sense that the set of conclusions that can be drawn from a set of sentences is not diminished if new sentences are added.)

Several researchers have proposed frameworks and techniques for nonmonotonic reasoning. McDermott and Doyle [37, 38] have developed a *nonmonotonic logic*. Reiter [46] has proposed inference rules (called *default rules*) whose applicability to a set of sentences Δ depends on what is *not* in Δ as well as what is. McCarthy [34] advocates the use of *circumscription* based on minimal models. Ginsberg [13] uses multiple (more than two) truth values to represent various degrees of knowledge. We will briefly describe one of these approaches, that based on minimal models, in order to illustrate what can be done. (See [47] for a thorough survey.)

Consider the general rule $(\forall x)\mathsf{Q}(x) \supset \mathsf{P}(x)$. We may know that this rule is not strictly correct without additional qualifications, and thus it cannot be included in a machine's knowledge base without making the knowledge base inaccurate. But we may want to use something like this rule to express the fact that "typically" all objects satisfying property Q also satisfy property P. Or we may want to use the rule in a system that can tolerate qualifications to be added later.

One way to hedge the rule (to avoid inaccuracy) is to introduce the concept of *abnormality*, denoted by the relation constant Ab [35]. Then we can say that all objects that are not abnormal and that satisfy property Q also satisfy property P:

$$(\forall x)\mathsf{Q}(x) \wedge \neg\mathsf{Ab}(x) \supset \mathsf{P}(x)\,.$$

Which objects are abnormal and which are not (if we know these facts) can be specified by other sentences in Δ. For example we may know that the objects denoted by A and B are abnormal: $\mathsf{Ab}(\mathsf{A}) \wedge \mathsf{Ab}(\mathsf{B})$.

If we do not know whether or not something, say the object denoted by C, is abnormal, we might be prepared to assume that it is not. Later, if we learn that it is, we can add Ab(C) to Δ.

How do we say that something is not abnormal unless it is required to be by what we have already said in Δ? One way to do this is to specify that the intended model lies within a special subset of the models of Δ. The subset is characterized by those models having the smallest possible sets of abnormal objects consistent with what we know must be abnormal. These models are called minimal with respect to Ab. McCarthy [34] has shown how to compute a sentence ϕ such that the models of $\Delta \wedge \phi$ are just the models in this subset. The formula ϕ is called the *circumscription formula* of Ab in Δ.

In the case in which the only objects that can be proved abnormal are those denoted by A and B, McCarthy's method (with an elaboration that allows the predicate P to "vary") calculates ϕ to be:

$$(\forall x)\mathsf{Ab}(x) \equiv [(x = \mathsf{A}) \vee (x = \mathsf{B})] .$$

If we additionally knew that $\mathsf{C} \neq \mathsf{B}$ and $\mathsf{C} \neq \mathsf{A}$, we could prove $\neg\mathsf{Ab}(\mathsf{C})$. Then, if Q(C), we could prove P(C).

Although the process of computing circumscription for general Δ is complex (ϕ might even be a second-order predicate calculus formula), there are some interesting special cases. Many of these have been investigated by Lifschitz [29] and are also described in [12].

5.2. Change

An intelligent machine must know something about how it perceives the world and about how its actions change the world. That is, it must know something about *see* and *effect*. The function *effect* characterizes how the world changes. If an agent is to perform appropriately in the world, it must be able to anticipate and influence these changes. Although it sounds unnecessarily tedious, an agent must know that after it moves from A to B, for example, it will be at B. It must also know that, all other things being equal, other objects will remain where they were before it moved to B. In summary, an agent must have as part of its knowledge some idea of how *effect* changes the world.

Several approaches have been pursued. Most work has been done using a conceptualization that includes objects called *states*. States are imagined as instantaneous snapshots of the world. Change is characterized as a transition from one state to another. Changes may occur as a result of actions on the part of the intelligent machine; we want our machines to be able to compute what actions they ought to perform in order that certain desirable states, called *goal states*, result. A key problem in characterizing the effects of a given machine action on the world involves how to specify which aspects of the world do not change. This has been called *the frame problem*.

The frame problem has been thoroughly treated in the literature. (See [7] and [44] for collections of articles. The latter collection includes several that discuss the problem from the standpoints of philosophy and cognitive psychology.) In attempting to deal with the frame problem in their system called STRIPS, Fikes and Nilsson [10] described the effects of a machine's actions by listing those relations that were changed by the action. They assumed that those relations not mentioned were not changed. Hayes [17, 18] introduced the notion of *histories* in an attempt to define a conceptualization in which the frame problem was less severe. McCarthy [35] and Reiter [46] proposed nonmonotonic reasoning methods for dealing with the frame problem. In the language of circumscription, their approaches assumed minimal changes consistent with the relations that were known to change. However, Hanks and McDermott [15] showed that a straightforward application of circumscription does not produce results strong enough to solve the frame problem. In response, Lifschitz [30] introduced a variant called *pointwise circumscription*. He also proposed reconceptualization of actions and their effects that permits the use of ordinary circumscription in solving the frame problem and the qualification problem [31]. Shoham [51] proposed an alternative minimization method related to circumscription, called *chronological ignorance*.

Although the frame problem has been extensively studied, it remains a formidable conceptual obstacle to the development of systems that must act in a changing world. This obstacle is faced by all such systems—even those whose knowledge about the world is represented in procedures. The designer of any intelligent machine must make assumptions (at least implicit ones) about how the world changes in response to the actions of the machine if the machine is to function effectively.

5.3. Uncertain knowledge

When one is uncertain about the world, one cannot specify precisely which relations hold in the world. Nevertheless, one might be able to say that at least one of a set of relations holds. Logical disjunctions permit us to express that kind of uncertain knowledge.

Logical representations (with their binary truth values) would seem to be inadequate for representing other types of uncertain knowledge. How do we say, for example, "It is likely that it will be sunny in Pasadena on New Year's day"? We could, of course embed probability information itself in the sentence, and this approach and others have been followed. Attempts to fuzz the crisp true/false semantics of logical languages have led to an active AI research subspecialty [23, 43, 53].

The approach followed by [41], for example, is to imagine that a probability value is associated with each of a set of possible conceptualizations (interpretations). The machine designer makes this assignment implicitly by composing a

set of first-order predicate calculus sentences each having a probability value. Each of these values is the sum of the probabilities of those interpretations that are models of the associated sentence. In this way the set of sentences with their probabilities induce constraints on what the probabilities of the interpretations can be. These constraints can then be used to calculate bounds on the probabilities of other sentences about the world. This approach can be computationally intractable, however, and many approximate methods for dealing with uncertain knowledge are being explored and used.

Just as the logical approach to AI provides a coherent framework in which many of the problems of designing intelligent machines can be clearly posed and understood, we expect that some appropriate combination of logic and probability theory will similarly aid our understanding of the problem of reasoning with uncertain information.

5.4. Embedded systems

> Those who will not reason
> Perish in the act:
> Those who will not act
> Perish for that reason. (W.H. Auden [4, p. 42].)

To be effective in their environments, machines must both *react* appropriately to sensory inputs in bounded time and *reason*, less hurriedly, about what to do. Since logical reasoning may, in general, require unbounded time, these two requirements place conflicting demands on the architecture of intelligent machines. Combining the two abilities in a seamless and elegant architecture is proving to be a difficult problem.

One might approach this problem either from the point of view of control theory, in which case one needs to add symbolic reasoning abilities to the inherent real-time characteristics of control systems; or from the point of view of AI, in which case one needs to find a way to achieve real-time performance in systems that are able to plan and reason. Very little research has been done on the connection between reasoning mechanisms and their environments. Instead, the need for real-time response and the supposed inefficiencies of logical reasoning have inspired a number of AI researchers to explore action-computation mechanisms that seem more related to control theory than to AI.

Rosenschein and Kaelbling [48] have proposed *situated automata*—finite state machines that do not reason with explicit sentences but, rather, react within a bounded time interval to sensory stimuli. These machines are compiled from declarative sentences specified by the designer. From the designer's point of view, the machines can be said to know the propositions represented by these sentences, but the sentences themselves are not explicitly represented or reasoned with in the machine. In this approach, logical reasoning is done at compile time, and the results of this reasoning are available at run time. (Even

though situated automata do not themselves perform explicit logical reasoning, some include them within the logical approach to AI because logic plays such an important role in their specification.)

Schoppers and Nilsson have each observed that in constructing a plan of actions the search process finds not only a path from the starting situation to a goal but finds several other paths as well. Ordinarily, these extraneous paths are discarded. Schoppers [50] suggests that with modest extra effort the paths from all possible starting states to a goal can be found during the plan-generation process. These paths can be stored as a *universal plan* that can be used to select appropriate actions in bounded time no matter how the unpredictable environment might throw the system off its current path. Nilsson [42] proposes that, in effect, multiple paths can be stored in a highly conditional plan structure called an *action network* consisting of a combinational circuit somewhat like those occuring in situated automata. Both universal plans and action networks share the virtue that actions appropriate to a wide range of environmental conditions can be selected in bounded time. Like situated automata, they pay for their improved runtime performance by investing extra time at compile or design time and by using extra space (to store the plan).

Other examples of architectures that do not rely (at run time at least) on the lumbering reasoning apparatus associated with explicit declarative representations can be found in the adaptive networks of the *connectionists* [49], in the finite-state machines of Brooks [6], in the PENGI system of Agre and Chapman [2], and in the *plan nets* of Drummond [9].

Good as they are for computing actions quickly however, systems that do no logical reasoning at run time may sometimes perish in the act. Imagine a Martian robot, for example, receiving a message from its Martian base that the base will not be able to refuel robots for five hours beginning in two hours. It would seem to be a reasonable strategy for robots to handle messages of this sort by receiving them as declarative sentences that are to be combined with other explicitly represented declarative knowledge and reasoned with to produce appropriate action. Anticipating what the appropriate responses should be for all possible such messages would seem to present untractable compilation and storage problems.

Connecting perception to reasoning and reasoning to action remains an important problem. For action nets and situated automata, for example, this problem becomes one of modifying the runtime system incrementally as new declarative knowledge is added to the system. Drummond [9] proposes the use of *situated control rules* generated by a planner to constrain the actions of a plan net. Kaelbling [22] has proposed hierarchical systems in which the lower levels are able to compute actions more quickly (if less intelligently) than the higher ones. Under time pressure, if the higher levels have not finished their more elaborate computations, the lower ones dominate.

I suspect that there are several other conceptual problems in connecting the

world to systems that reason with explicitly represented sentences. As a columnist recently wrote [45] ". . . I sometimes worry about the men I know who live alone. I picture some of them thinking out loud about a ham and Swiss on rye, and then just sitting there, slowly starving to death. There's some synapse missing, the one that links desire and action." Transforming the highly indexical, situation-dependent information gleaned from sensors into context-free representations, and then converting information in those representations back into context-sensitive action, likely will involve "synapses" that have not yet been adequately considered by the logicists.

6. Conclusions

Logic provides the vocabulary and many of the techniques needed both for analyzing the processes of representation and reasoning and for synthesizing machines that represent and reason. The fact that we discover elements of reasoning and intelligent behavior that challenge the techniques of ordinary logic is no reason to abandon the solid base that we have already built. On the contrary, it appears that imaginative extensions to ordinary first-order logic are successfully dealing with many of its purported inadequacies.

Logic itself (originally invented to help formalize and explain human reasoning) has evolved dramatically in the last 2000 years. Anyone who attempts to develop theoretical apparatus relevant to systems that use and manipulate declaratively represented knowledge, and does so without taking into account the prior theoretical results of logicians on these topics, risks (at best) having to repeat some of the work done by the brightest minds of the last two millenia and (at worst) getting it wrong!

Of course, it may ultimately turn out that we cannot adequately and usefully represent knowledge needed by intelligent machines in declarative sentences. If we cannot, each application would seem to require a machine most of whose knowledge has to be designed in to procedures specialized to the application. Many of these specialized systems would have similar or identical bodies of knowledge, but even if largely identical, each body would have to be separately and specially (and thus expensively) crafted and re-crafted for each niche system. There is good reason to recoil from that prospect. There are too many niches! Considering the high payoff, the impressive results obtained so far, and the lack of promising alternatives, the grand vision of the logicists appears to me to be the approach of choice.

Acknowledgement

The author thanks the Palo Alto Laboratory of the Rockwell Scientific Center for support. Several people provided helpful suggestions, including Jon

Doyle, Michael Genesereth, Carl Hewitt, David Kirsh, Jean-Claude Latombe, John McCarthy, Devika Subramanian, Yoav Shoham, and many Stanford graduate students.

References

[1] E. Abbey, *Desert Solitaire* (Ballantine, New York, 1971).
[2] P. Agre and D. Chapman, Pengi: An implementation of a theory of activity, in: *Proceedings AAAI-87*, Seattle, WA (1987) 268–272.
[3] D.E. Appelt, *Planning English Sentences* (Cambridge University Press, Cambridge, 1985).
[4] W.H. Auden, *Collected Shorter Poems: 1927–1957* (Random House, New York, 1966).
[5] R. Brachman, V. Gilbert and H. Levesque, An essential hybrid reasoning system: knowledge and symbol level accounts of KRYPTON, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985).
[6] R.A. Brooks, Intelligence without representation, *Artif. Intell.* **47** (1991) 139–159, this volume.
[7] F. Brown, ed., *The Frame Problem in Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1987).
[8] P.R. Cohen and H.J. Levesque, Intention is choice with commitment, *Artif. Intell.* **42** (1990) 213–261.
[9] M. Drummond, Situated control rules, in: J. Weber, J. Tenenberg and J. Allen, eds., *Proceedings of the Rochester Planning Workshop: From Formal Systems to Practical Systems*, Rochester, NY (1988).
[10] R.E. Fikes and N.J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artif. Intell.* **2** (1971) 189–208.
[11] M.R. Genesereth, The use of design descriptions in automated diagnosis, *Artif. Intell.* **24** (1984) 411–436.
[12] M.R. Genesereth and N.J. Nilsson, *Logical Foundations of Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1987).
[13] M. Ginsberg, Multivalued logics: a uniform approach to inference in artificial intelligence, *Comput. Intell.* **4** (1988).
[14] B.J. Grosz, D.E. Appelt, P.A. Martin and F.C.N. Pereira, TEAM: An experiment in the design of transportable natural-language interfaces, *Artif. Intell.* **32** (1987) 173–243.
[15] S. Hanks and D. McDermott, Default reasoning, nonmonotonic logics, and the frame problem, in: *Proceedings AAAI-86*, Philadelphia, PA (1986).
[16] P.J. Hayes, In defence of logic, in: *Proceedings IJCAI-77*, Cambridge, MA (1977) 559–565.
[17] P.J. Hayes, The naive physics manifesto, in: D. Michie, ed., *Expert Systems in the Micro-Electronic Age* (Edinburgh University Press, Edinburgh, 1979) 242–270.
[18] P.J. Hayes, The second naive physics manifesto, in: J.R. Hobbs and R.C. Moore, eds., *Formal Theories of the Commonsense World* (Ablex, Norwood, NJ, 1985) 1–36; also in: R. Brachman and H. Levesque, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, San Mateo, CA, 1985).
[19] B. Hayes-Roth, A blackboard architecture for control, *Artif. Intell.* **26** (1985) 251–321.
[20] J.R. Hobbs, ed., Commonsense Summer: final report, Rept. CSLI-85-35, Center for the Study of Language and Information, Stanford University, Stanford, CA (1985).
[21] J.R. Hobbs and R.C. Moore, eds., *Formal Theories of the Commonsense World* (Ablex, Norwood, NJ, 1985).
[22] L.P. Kaelbling, An architecture for intelligent reactive systems, in: M. Georgeff and A. Lansky, eds., *Reasoning about Actions and Plans* (Morgan Kaufmann, San Mateo, CA, 1987) 395–410.
[23] L.N. Kanal and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence* (North-Holland, Amsterdam, 1986).

[24] K. Konolige, Belief and incompleteness, in: J.R. Hobbs and R.C. Moore, eds., *Formal Theories of the Commonsense World* (Ablex, Norwood, NJ, 1985) 359–404.
[25] D.B. Lenat, The nature of heuristics, *Artif. Intell.* **19** (1982) 189–249.
[26] D.B. Lenat, Theory formation by heuristic search. The nature of heuristics II: Background and examples, *Artif. Intell.* **21** (1983) 31–59.
[27] D.B. Lenat, EURISKO: A program that learns new heuristics and domain concepts. The nature of heuristics III: Program design and results, *Artif. Intell.* **21** (1983) 61–98.
[28] D.B. Lenat and E.A. Feigenbaum, On the thresholds of knowledge, *Artif. Intell.* **47** (1991) 185–250, this volume.
[29] V. Lifschitz, Computing circumscription, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 121–127.
[30] V. Lifschitz, Pointwise circumscription: preliminary report, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 406–410.
[31] V. Lifschitz, Formal theories of action, in: F. Brown, ed., *The Frame Problem in Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1987) 35–57.
[32] J. McCarthy, Programs with common sense, in: *Mechanisation of Thought Processes, Proc. Symp. Nat. Phys. Lab.* **1** (Her Majesty's Stationary Office, London, 1958) 77–84; also in: M. Minsky, ed., *Semantic Information Processing* (MIT Press, Cambridge, MA, 1968) 403–410; and in: R. Brachman and H. Levesque, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, San Mateo, CA, 1985).
[33] J. McCarthy, Ascribing mental qualities to machines, Tech. Rept. STAN-CS-79-725, AIM-326, Department of Computer Science, Stanford University, Stanford, CA (1979).
[34] J. McCarthy, Circumscription—a form of non-monotonic reasoning, *Artif. Intell.* **13** (1980) 27–39; also in: B.L. Webber and N.J. Nilsson, eds., *Readings in Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1982).
[35] J. McCarthy, Applications of circumscription to formalizing commonsense knowledge, *Artif. Intell.* **28** (1986) 89–116.
[36] J. McCarthy, Mathematical logic in artificial intelligence, *Daedalus: J. Am. Acad. Arts Sci.* (1988) 297–311.
[37] D. McDermott and J. Doyle, Non-monotonic logic I, *Artif. Intell.* **13** (1980) 41–72.
[38] D. McDermott, Non-monotonic logic II: Non-monotonic modal theories, *J. ACM* **29** (1982) 33–57.
[39] D. McDermott, A critique of pure reason (with peer commentaries), *Comput. Intell.* **3** (1987) 151–237.
[40] R.C. Moore, A formal theory of knowledge and action, in: J.R. Hobbs and R.C. Moore, eds., *Formal Theories of the Commonsense World* (Ablex, Norwood, NJ, 1985).
[41] N.J. Nilsson, Probabilistic logic, *Artif. Intell.* **28** (1986) 71–87.
[42] N.J. Nilsson, Action networks, Unpublished working paper, Stanford University, Stanford, CA (1988); also in: J. Weber, J. Tenenberg and J. Allen, *Proceedings of the Rochester Planning Workshop: From Formal Systems to Practical Systems*, Rochester, NY (1988).
[43] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
[44] Z. Pylyshyn, *The Robot's Dilemma: The Frame Problem in Artificial Intelligence* (Ablex, Norwood, NJ, 1987).
[45] A. Quindlen, Life in the 30's, *The New York Times* (May 5, 1988).
[46] R. Reiter, A logic for default reasoning, *Artif. Intell.* **13** (1980) 81–132.
[47] R. Reiter, Nonmonotonic reasoning, in: *Annual Reviews of Computer Science* **2** (Annual Reviews, Palo Alto, CA, 1987) 147–186.
[48] S.J. Rosenschein and L.P. Kaelbling, The synthesis of machines with provable epistemic properties, in: J.Y. Halpern, ed., *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge* (Morgan Kaufmann, San Mateo, CA, 1986) 83–98.
[49] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* **1**: *Foundations*; **2**: *Psychological and Biological Models* (MIT Press, Cambridge, MA, 1986).
[50] M.J. Schoppers, Universal plans for reactive robots in unpredictable domains, in: *Proceedings IJCAI-87*, Milan, Italy (1987).

[51] Y. Shoham, *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence* (MIT Press, Cambridge, MA, 1988) Chapter 4.

[52] Y. Shoham and N. Goyal, Temporal reasoning in artificial intelligence, in: H. Shrobe, ed., *Exploring Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1988) 419–438.

[53] E.H. Shortliffe, *Computer-Based Medical Consultations: MYCIN* (Elsevier, New York, 1976).

[54] B.C. Smith, The owl and the electric encyclopedia, *Artif. Intell.* **47** (1991) 251–288, this volume.

[55] P. Smolensky, On the proper treatment of connectionism, *Behav. Brain Sci.* (with peer commentary) **11** (1988) 1–74.

[56] M.E. Stickel, A nonclausal connection-graph resolution theorem-proving program, in: *Proceedings AAAI-82*, Pittsburgh, PA (1982) 229–233; also in: M. Stickel, A nonclausal connection-graph resolution theorem-proving program, SRI AI Center Tech. Note 268, SRI, Menlo Park, CA (1982).

[57] M.E. Stickel, A PROLOG technology theorem prover, in: *Proceedings IEEE Symposium on Logic Programming*, Atlanta City, NJ (1984) 211–217.

[58] R. Weyhrauch, Prolegomena to a theory of mechanized formal reasoning, *Artif. Intell.* **13** (1980) 133–170; also in: B.L. Webber and N.J. Nilsson, eds., *Readings in Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1982).

[59] S. Winker, Generation and verification of finite models and counterexamples using an automated theorem prover answering two open questions, *J. ACM* **29** (1982) 273–284.

[60] T. Winograd, Frame representations and the declarative/procedural controversy, in: D. Bobrow and A. Collins, eds., *Representation and Understanding: Studies in Cognitive Science* (Academic Press, New York, 1975) 185–210; also in: R. Brachman and H. Levesque, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, San Mateo, CA, 1985).

[61] L. Wos, S. Winker, B. Smith, R. Veroff and L. Henschen, A new use of an automated reasoning assistant: open questions in equivalential calculus and the study of infinite domains, *Artif. Intell.* **22** (1984) 303–356.