HECTOR J. LEVESQUE

LOGIC AND THE COMPLEXITY OF REASONING*

What does logic have to do with computational approaches to the study of cognition? Very little, it would seem. For one thing, people (even trained logicians) are unquestionably very bad at it, compared to their skill at (say) reading or recognizing tunes. And computers have a hard time with it too: the computational activity that goes with logic, theorem proving of some sort, appears to be computationally intractable. Given its apparent difficulty, it seems quite unlikely that logic could be at the root of normal, everyday thinking. What I want to suggest here, however, is that rather than closing the book on logic (except perhaps as a mathematical tool for theoreticians so inclined), these facts force us into a less idealized view of logic, one that takes very seriously the idea that certain computational tasks are relatively easy, and others more difficult. In other words, it leads us to consider the *computational complexity* of logic in its application to cognition.[1]

Perhaps the most striking and puzzling fact about cognition from a computational standpoint is how it can be so *easy* in the presence of such a vast number of potentially relevant beliefs. How is it, for example, that we can effortlessly answer questions like:

What would surfing be like in Saskatoon?

Could a crocodile run a steeplechase?

Should baseball players be allowed to glue small wings onto their caps?

We obviously do not simply remember (or perceive) the answers. And although we have never thought about such ridiculous things before, we clearly have the beliefs necessary to answer the questions. But how exactly do we put them together, and follow a train of thought without getting derailed?

Journal of Philosophical Logic **17** (1988) 355–389.
© 1988 *by Kluwer Academic Publishers.*

Before we can explain why this is effortless, of course, we need to establish a measure of difficulty. Surprisingly, perhaps, complexity theory in computer science has had very little impact on cognitive science.[2] What I want to do here is to show how complexity theory provides necessary constraints and unifying principles for a computationally-based cognitive science. First, I will review very quickly the idea of a computational system and why I think complexity considerations are so important. I will then present a research strategy that has complexity at the forefront, and answer briefly two common objections to it. To make things concrete, I will then focus on a type of computational system (called a knowledge-based system) that has a logical substrate, and present a number of specific ways that the computational complexity constraints can be addressed. Overall, this will not form anything like an airtight argument for this way of looking at things. But on the other hand, I do not believe that the approach stands or falls on any of the individual points I will make. The more relevant concern, perhaps, is to what extent the approach is right- or wrong-headed.

1. COMPUTATIONAL SYSTEMS

One way to think of cognitive science is that it starts with a picture something like that of Figure 1: we have a creature that lives within a complex environment, and what we want to do is account for how this situated creature (to use the West Coast buzzword) is able to act and react to its environment in the very general, flexible, and adaptable way that we do. Obviously, there are different ways of approaching this – for example, in terms of evolution, or in terms of biology, or ecology, or anthropology, or whatever. But one very popular way of explaining how this kind of intelligence can exist at all is in terms of computation.

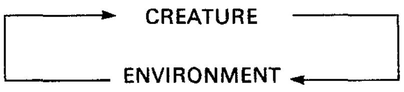

Fig. 1.

What does computer science have to offer here? We certainly do not expect the creature to be built at all like an electronic computer (although someday this might be true). The brain is, after all, no more like a current electronic computer than it was like a wind-up clock, a steam engine, or a telephone switchboard. The idea rather is to focus not on what the brain (or the central nervous system) *is*, but on what it *does*. This could be done, for example, at the level of neuronal activity (at one extreme), or in statistical terms over populations of neurons (at the other). But there is a third intermediate possibility offered by computer science: we can look at what the brain does as a form of computation. The key insight is that the brain, like certain collections of electronic components, certain arrangements of gears and motors, certain configurations of pipes and valves, certain organizations of messengers and messages, and so on, can be usefully analyzed as a computational system.[3]

What do I mean by a computational system? Very roughly, we can view the above picture broken down as in Figure 2:

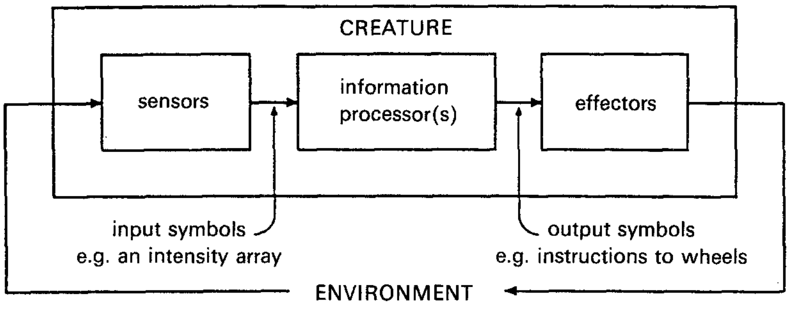
CREATURE
sensors
information processor(s)
effectors
input symbols e.g. an intensity array
output symbols e.g. instructions to wheels
ENVIRONMENT

Fig. 2.

there are *sensors* that are able to convert energy from the environment into some sort of symbolic form (here, light energy is converted into an array of intensity values);[4]

there are *effectors* that are capable of converting symbolic structures into changes to the environment (here, instructions to wheels are converted into robot motion); and in between,

> there is some sort of general *information processor* that is able to produce a stream of output symbols given a stream of input symbols, for example, to decide what instructions to the wheels (or to other effectors) are appropriate given the array of intensity values (or other sensory input).

I suspect that nobody in cognitive science believes that an analysis at this level will be sufficient to explain all that can be explained about this creature's cognition (any more than computer scientists would feel that this model exhausts what there is to say about electronic computers); nonetheless, there is substantial agreement that this kind of breakdown (with the constraints discussed below) is very instructive. And for certain kinds of activity that the creature engages in (using language, making complex decisions, solving problems, *etc.*), it appears to be the only hunch we have of how they could be physically realized.

What constraints are imposed by saying that a system is computational? What transductions from inputs to outputs are indeed computational?[5] If you have never written a program before, there seems to be no limit to what computers can be made to do. There would be little surprise if it were generally announced that computers were able to converse in unrestricted English, for example. Can't they already? Once you have written a few programs, you gradually come to the opposite conclusion: computers are hardly able to do *anything*, and what they can do has to be based on unbelievably explicit instructions from the programmer. You can't just tell a computer to focus on the relevant inputs, or to allow for possible noise, or to do something until it feels right, unless you can unpack these notions in terms of formal symbolic operations. They are not the kind of primitive operations out of which computational systems are built.[6]

After a programmer has acquired a bit of experience, she comes to realize that even within the confines of what appeared at first to be pathologically detailed symbol manipulation, there is quite a bit of elbow room. Computational systems can indeed be built (with some effort) to play chess, or to find edges in pictures.[7] However, it also becomes clear that not all formal symbolic operations can be incorporated within real computational systems. Under the usual representations, determining if a sentence is a theorem of the predicate calculus, or if a Turing machine will halt on a given input are problems that cannot be solved in these terms. And among those that can, many of them cannot be performed in anything like reasonable time, "high-speed computer" notwithstanding. So, for example, checking if two graphs have the same structure, or determining if an arbitrary sentence of the propositional calculus is a tautology, are formally specifiable and computable operations that (apparently) cannot be completed within tolerable resource bounds.[8] If we want the notion of a computational system to explain anything, we must reduce the activity in question to symbolic operations that can indeed be carried to completion within acceptable resource bounds. It is this last constraint that I wish to examine here.

2. RESOURCE BOUNDS

The first observation about this notion of resource bounds is that it is not just a question of "efficiency," a secondary property like the colour of a car or the weight of a telephone. If we propose a model of a cognitive activity in terms of a computational task with an exponential lower bound, and the model cannot limit the size of the input, it tells us *nothing* about how the cognitive activity is physically possible.[9] It is simply not realistic, even if it is computational, strictly speaking. To assume, as an idealization, that a machine can run sufficiently quickly to get such a task done is not very different from assuming, again as an idealization, that the machine can solve the halting problem.

The above concerns are what I take to be the principal motivation behind the great interest in the massively parallel "connectionist" architectures (Rumelhart and McClelland, 1986). Given a neuron firing rate of less than 1000 Hz, and the fact that many recognition tasks are completed in under 100 milliseconds (Ballard, 1986), any analysis that requires more than 100 sequential time steps is not biologically realistic. The issue for the connectionists, then, is what operations *can* be performed effectively within these resource bounds, given about 10^{11} neurons and 10^{15} connections among them to work with?

But there are two ways at least to approach this question. One can propose a specific computational architecture involving a certain structure (for example, a graph with weights on edges, with certain nodes designated as inputs, others as outputs) and operations over it (replacing the value at each node by some linear function of the previous value and the value at the neighbouring nodes). One then asks: What can such architectures compute? How long do they take? and so on. The second strategy involves looking at the information processing (or IP) tasks themselves as mappings from streams of input symbols to streams of output symbols, and asking questions like: What are the properties of such tasks? Which of them are cognitively useful? physically realizable?

Now I take these to be two different research strategies for dealing with the same question, and each has advantages and disadvantages. I intend to focus on the second strategy here, but not to the exclusion of the first. The two points of view are complementary, in my opinion. In particular, nothing I have said excludes IP tasks that involve probabilities, partial matches, combining evidence, degrading gracefully, and many of the other things emphasized by connectionists.[10]

3. TURNING MACHINE COMPLEXITY

What does it mean for an information processing task to be physically realizable? To a first approximation, this means that it can deal with its inputs within tolerable resource bounds. I will take up the issue of what inputs need to be allowed for below. As for tolerable resource bounds, what we need is a machine that is fast enough to adapt to and exploit its environment. Even if it is possible in principle to figure out that the thing coming towards you is a truck, you need to be able to get to this conclusion before the truck gets to you.

The complication here is that in the second research strategy, we are looking at IP tasks themselves independently of any architecture. We need to be careful before deciding that something is physically impossible. We must allow for all sorts of technological developments (much faster machines, for example); we must allow for massive parallelism, in its current and future incarnations; we must also allow for clever algorithms, and very, very different ways of performing the same task. The difficulty is that this seems to lead us right back to the first research strategy, that we need to make architectural and algorithmic assumptions to get any hard constraints.

But there is a way to sidestep these difficulties, if we are willing to be more modest in our analysis of physical realizability. First, we can look at IP tasks in terms of the number of operations required as a function of the size of the input that needs to be considered.[11] Next, we can be satisfied with sorting the tasks into two categories: the physically unrealizable and the physically *plausible*. Those in the first category can be eliminated from further consideration as part of a computational model of cognitive activity, and those in the second category are tasks that we will be prepared to examine further, perhaps (though not necessarily) using specific computational architectures.[12]

The claim here is that it is indeed possible to usefully sort IP tasks in this way without any assumptions about how the tasks will be executed. The starting point for this claim is the Church-Turing thesis, which in its strongest form is something like:

CHURCH-TURING THESIS. Anything that can be computed at all can be computed by a Turing machine.

This thesis is by now almost universally believed. It does not address the issue of resource bounds, however. But a second one does:

TURING COMPLEXITY THESIS. Anything that can be computed at all can be computed by a Turing machine *with at most a polynomial slowdown.*

Although I have never seen the thesis formulated quite like this, it is nonetheless almost universally believed by computer scientists as well. It is also a much stronger claim, since it asserts that all computers work at roughly the same speed as a Turing machine (to within a polynomial of the input size).[13] In particular, it has this as a consequence:

COROLLARY. *Anything that a Turing machine cannot calculate in polynomial time cannot be calculated* at all *in polynomial time.*

If we let P stand for the class of all tasks that Turing machines can perform in polynomial time, then the corollary says that the tasks outside of P cannot be performed in polynomial time by anything physical. Thus, *if* the inputs to these tasks cannot be bounded (and this is a big if), then they are simply not physically realizable. Again, this is not to say that just because a task is in P that it *is* realizable or that a (sequential) Turing machine is a reasonable way to calculate it. But we can use P as the category of tasks that are physically plausible, and know that unlike other possible categories, it is robust enough that tasks in $\bar{P}$ are not plausible for *any* architecture, past, present, or future. Moreover, if we believe, as most computer scientists do, that $P \neq NP$, then we believe that tasks that are NP-hard are also physically unrealizable, and cannot be part of a realistic computational model of cognitive activity.

4. TWO OBJECTIONS

The argument I have presented so far is rough – probably much rougher than what this journal is accustomed to. It is not my intention to defend the research strategy much further, since ultimately we should look to the pudding for the proverbial proof. However, there are at least two objections that are so common that they should be dealt with, if only in a somewhat cursory fashion.

The first objection goes something like this:

> Anything even remotely interesting from a cognitive standpoint is NP-hard. For example, *polyhedral scene labelling* was shown in (Kirousis and Papadimitriou, 1985) to be NP-complete. But this is a tiny part of vision, and for us, vision is precise, and effortless. So NP-completeness does not measure the real tractability of a task, and the classification itself is not very useful.

I want to make two points. First of all, not everything of interest is NP-hard (see Sections 7–12). Secondly, when assessing the relevance of intractability results, we need to consider carefully the admissible inputs. What exactly *can* people do effortlessly? How many lines do we expect in a scene? Certainly, not as many as the number of independent beliefs (by whatever measure) that a person might hold, for example. Can people handle unrestricted drawings? In particular, the complexity result was derived by encoding formulas of the propositional calculus as polyhedral line drawings that could be labelled only if the corresponding formulas were satisfiable. It would be very, very, surprising if anybody (or anything) was able to label such drawings "precisely and effortlessly." Because it can be reduced to a logical puzzle, scene labelling is much more difficult in general than (say) answering a question about crocodiles running steeplechases, regardless of the amount of potentially relevant crocodile knowledge that would have to be taken into account.[14] It is a mistake to take a problem that people can handle readily in its simplest form, and then to look at the complexity of dealing with a much more general form. Although some idealization is perhaps unavoidable, we must guard against attributing to people overpowering computational abilities, if we are to preserve any psychological relevance.

The second objection deals directly with the issue of inputs:

> What are these "admissible" inputs? We are never going to be able to predict beforehand what our systems will need to be able to deal with. How many edges in a scene? How noisy a signal? How nested a sentence of English? The problem with complexity analyses is that they deal with *worst case* behaviour, and these worst cases may never arise in practice. So these analyses tend to be overly pessimistic about what is or is not realizable. Besides, the whole idea of "handling inputs" is too restrictive. Situations are dealt with more or less well, and we should not expect perfection of our computational models. People do make mistakes, lose interest, and sometimes even give up when things get too complicated or take too long. There need to be resource bounds on our models, of course, but we may not be able to predict in advance when they will be necessary. Indeed, realistic cognitive activity is much too complex for any sort of "neat" *a priori* mathematical analysis in terms of inputs and outputs.

There are really two objections here, and I will take up the second one (regarding what we should expect of computational models) first. The assumption I am making here is that we are interested in explaining cognitive activity, and that properties of a computational model that are not designed to correspond to the activity being modelled are of no interest. The model has no instrinsic value by itself, and is only of interest insofar as it explains the behaviour in question.[15] The point is that this remains true even if this behaviour includes memory limitations, attention lapses, or anything else that might be viewed as less than ideal. This is all part of the subject matter, and needs to be modelled deliberately as part of an IP task. Even if the behaviour is "messy," the model itself cannot be. We may or may not choose to model idealized cognitive activity, and the further away we move from idealization the more difficult our modelling job may be, but this decision does not alter the hard fact that a model that has the wrong behaviour or the right behaviour for unaccountable reasons has no explanatory power.

Given this, if we want to deal with mistakes, memory limitations, and the like, we should do so as carefully and as deliberately as any other feature of the cognitive behaviour we wish to understand. Consider, for example, this notion of "giving up" on cognitive tasks. This is, I feel, somewhat misleading. For one thing, I believe it only happens when we are in a certain *puzzle mode*, as I will discuss again below, dealing with a problem by deliberately following a specific procedure. But even in such cases, we can *always* reliably decide if we know what to do next.[16] In other words, when it comes to problem solving, although people may choose to stop working on a problem, they do not "time out." Any model that suggests that we are *uncontrollably* booted to an executive level once cognitive resources are exhausted is just plain wrong.[17] This would be an instance of what I mean by not taking an issue (in this case, resource limitations) seriously.

Similar considerations apply to other ways cognitive behaviour can be less than ideal. We may contend, for example, that this behaviour involves approximations (in some technical sense) or graceful degradations (perhaps dynamically estimating how much effort to spend) to some idealized, correct behaviour. If that is the case, we should take these approximations seriously and focus on them as IP tasks in their own right, and ask what computational characteristics they have (for example, what their worst-case complexity properties are).

I have argued elsewhere about why we should care about worst cases of some sort (instead of (say) average cases) (Levesque, 1986a). In a nutshell, we need to establish that our models will scale up properly and continue to correspond in an appropriate way to the cognitive activity being modelled for the full range of inputs under investigation. If the cognitive activity itself degrades itself in some way on large inputs or in certain situations, fine; but the *correspondence* between the model and this activity should never degrade or depend on the benevolence of the environment. The true test of the robustness of the model is indeed how well it does in the most extreme cases.[18] Interestingly enough, it is also true that the computational tasks that have admitted algorithms that work well *in practice* have been precisely those with provably good worst-case properties (that is, in P).[19]

But what about this notion of admissible inputs and the fact that worst cases can be overly pessimistic? Here, I think the objection is valid. It is often very difficult to decide what is or is not a potential input. What two dimensional arrays of intensity values correspond to scenes that occur naturally? The best we can hope for is a characterization that does not admit too many implausible cases.

On the other hand, it should be noted that intuitions about what is or is not computationally tractable are often very wrong. By looking at proofs of intractability for a perhaps overly general set of inputs, a theorist can often see what features of the input set combine to produce the difficulty, and then go on to examine input sets where these features are *not* present. If often happens that cases where the proof of intractability fails can be proven to be tractable. For example, in the polyhedral scene labelling discussed above, Kirousis and Papadimitriou identify a class of scenes (called orthohedral) that are provably tractable. A similar result appeared in work I did with Ron Brachman (Brachman and Levesque, 1984) involving description matching. The general case of the problem is NP-hard, but in doing the proof, it was possible to isolate a special case (where there is no "role differentiation") where the description matching is provably tractable. By restricting ourselves to these cases, many unrealistic inputs are eliminated, and we end up with much less pessimistic worst cases. If these cases are then seen as too limited to be cognitively useful, the theorist can attempt to generalize as appropriate, and the process iterates.

5. KNOWLEDGE-BASED SYSTEMS

To make a lot of this discussion more concrete, I will focus on a particular type of computational system called a *knowledge-based system.*[20] Going back to Figure 2, a knowledge-based system is just a particular way of structuring an information processor. The main refinement here is that the information processor traffics in very special internal symbols. By this I mean that the processor is divided into further components that have these symbols as their inputs or outputs. The symbols are distinguished by the fact that we, from the outside, can understand them *propositionally* and, because of the way the system behaves overall, as standing for its beliefs.[21] The components of the information processor are (see Figure 3):

> a *perception* component at one end that converts sensory input symbols into appropriate sentences about the world (such as: *There is a truck coming at me*);
>
> an *action* component that converts sentences about what ought to be done (such as: *I should get out of the way*) into instructions to appropriate effectors; and in between,
>
> a *knowledge representation and reasoning* component that manages these sentences over time and mediates the connection between perception and action.

One would expect the connection between the reasoning component and the other two to be bi-directional: perception must be sensitive to the beliefs of the system, and in deciding how to act, new beliefs will normally be generated.[22] So the task of the knowledge representation and reasoning component of a knowledge-based system, and the task that I wish to focus on, is this: given over time a large collection of sentences representing the beliefs of the system, extract from these

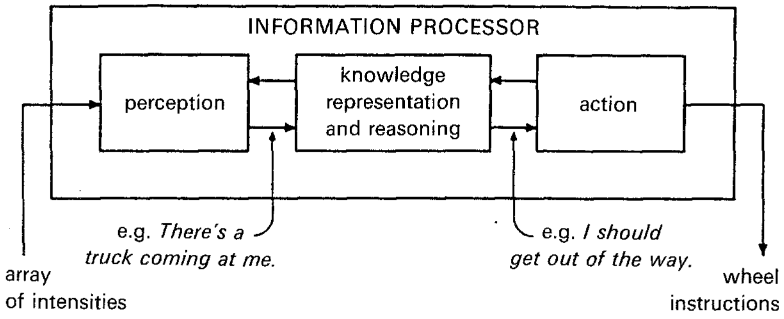
INFORMATION PROCESSOR
perception
knowledge representation and reasoning
action
e.g. There's a truck coming at me.
e.g. I should get out of the way.
array of intensities
wheel instructions

Fig. 3.

sentences information that is relevant to the future action and perception of the system.[23]

Before looking at this task, it is perhaps worthwhile reviewing why anyone would think that the knowledge-based system structure is a reasonable one for computational systems.[24] A more detailed (and convincing) argument can be found in (Pylyshyn, 1984).

The main argument *against* these systems is that given a certain overall behaviour, it is usually possible to achieve it otherwise, and with much better performance. It is often noted, for example, that as we get better at doing something, we apparently *think* about it much less. A good example is playing pinball: thinking about what your fingers are doing is a good way to lose the ball. Similar considerations seem to apply to driving a car (Dreyfus and Dreyfus, 1986). The question, of course, is how far this goes: what about playing chess? getting lunch? staying alive??

The principal advantage I see in a knowledge-based system involves very open-ended tasks. The design allows us to structurally isolate *reasons* for the behaviour, in the beliefs that are held. We often say, for example, that we did so-and-so *because* we thought that such-and-such. Much of this, no doubt, is rationalization after the fact.[25] But in so very many cases, an explanation involving belief is the one that captures just the right contingencies.

Consider, for example, reasons for answering the phone. Clearly, nothing that can be *observed* (a ring, a buzz, a flashing light, a message from the guy down the hall) is by itself sufficient if you have the right beliefs (as when you believe that the phone is being tested or that the call is a practical joke). But what does seem to be both necessary and sufficient in all cases is that you form an appropriate belief of some sort, for example, that there is somebody on the phone that you choose to talk to.

A knowledge-based system, then, is one where the connection between belief and behaviour is not mere rationalization: the architecture of the system is such that the presence of specific symbolic structures results in appropriate behaviour. Because what causes this behaviour has *not* been distributed throughout the system as a whole, this modular design is well suited to updating its information (or finding new uses for it). Indeed the hallmark of a knowledge-based system is that it can readily be *told* facts about the world it lives in, and adjust its overall behaviour accordingly. Imagine, for example, trying to raise canaries or to make money with rare coins by trial and error, and compare this to first reading a book on the subject. Overall, there seems to be a clear evolutionary advantage in being able to exploit an oral and written culture in a direct way, and this appears to require the ability to maintain a large collection of beliefs about the world, in a way that is removed from the immediate concerns of perception and action. As Pat Hayes puts it in (Hayes, 1987), "This sort of general awareness of the world around us, and general ability to talk about it, to learn about it from observation and language, and to act on the basis of facts no matter from where they come, seems to me to be impossible to explain without some assumption which amounts to a general-purpose store of information held in our heads." Or this, at least, is the hypothesis.

Getting back to the knowledge representation and reasoning task itself, there are again at least two ways to proceed. We can examine specific architectures (such as resolution theorem provers, production systems, or parallel semantic networks) and study their properties at what Newell has called the symbol level (Newell, 1982). Alternatively, we can look at the task itself in terms of information content at what Newell calls the knowledge level: what can be surmised about the world given the current set of beliefs (which we will call the *knowledge base*, or KB)? How we should do this is a very open question at this point. We will obviously want to use any facts, laws, rules, or definitions in the KB; but we may also want to use prototypical cases, defaults, generalizations, rules of thumb, hunches, and other educated and uneducated guesses.

The task, however, has a fairly clear base case: we can ask what *must* hold by virtue of the rules of the KB language. In other words, the task is to determine what is *entailed* by the KB. This brings *logic* into the picture, since on at least one reading, logic *is* the study of entailment in this sense.

6. LOGIC AND REASONING

This base case may or may not be cognitively useful, but it does present a very serious difficulty which can be summarized (somewhat glibly) as follows:

1. We need negation and disjunction facilities in our KB language (Moore, 1982).
2. With them, calculating entailments is computationally intractable, assuming P $\neq$ NP, (Cook, 1971).
3. Things are worse with quantifiers (Church, 1936).

It looks as if this whole approach is a waste of time. Minsky has been quoted as saying that logic is fine, just *bad psychology*: it is not what people do. But even if it was good psychology, what I am suggesting here is that as it stands, it is simply too demanding computationally.

There are two obvious reactions to this situation. You can drop down to the symbol level, look at specific architectures, and ask exactly what *can* we build? What will work? Or, you can stay at the knowledge level, and ask a more subtle and tricky question: what information processing tasks of this type are both logically meaningful and computationally realizable? How can the reasoning task be varied and manipulated to sidestep the above difficulties?

My thesis is as follows: The deviations from classical logic that will be necessary to ensure the tractability of reasoning stand in very close correspondence to the deviations from logic that we would have to make anyway to be psychologically realistic. If we look at the kinds of mistakes people make, the kinds of problems people run into, and the corners that are cut to get around them, we will find modifications to classical logic that ensure the computational tractability of the associated thinking.

To see how this might work, it is important to first distinguish between how formal logic is used in knowledge representation, and its intended use in mathematics.[26] Consider for example, the following sentences of the predicate calculus:

1. $\neg \mathrm{In}(\mathrm{block1}, \mathrm{box})$
2. $\mathrm{In}(\mathrm{block1}, \mathrm{box}) \vee \mathrm{In}(\mathrm{block2}, \mathrm{box})$
3. $\exists x[\mathrm{Noisy}(x) \wedge \mathrm{In}(x, \mathrm{box})]$
4. $\forall x[\mathrm{In}(x, \mathrm{box}) \supset \mathrm{Light}(x)]$

The first one says that Block 1 is not in the box, *but does not say where it is*; the second one says that either Block 1 or Block 2 is in the box, *but does not say which*; for the third one, imagine picking up the box, shaking it and saying that there is something noisy in the box, *without saying what it is*; and finally, the last one says (again after picking up the box, for instance) that everything in the box is light, *without saying what those things are.*

The thing to notice about these examples is that the connectives of first-order logic are not being used to express properties of infinite sets as they might be in the foundations of mathematics. Instead, they are being used to represent incomplete knowledge about the world, knowledge that does not nail down all the specifics of the situation in question. Typically, they provide the ability to say that one of some number of conditions holds without having to say which. As far as knowledge representation is concerned, the representational expressiveness of a language like that of first-order logic is not so much in what it allows you to say, but in what it allows you to *leave unsaid.*

What does this have to do with tractability? The point is this: The more that is left unsaid, the more possibilities are allowed by what is said. To determine what is *entailed* by what is said, all of these possibilities have to be covered one way or another. Proof methods may differ on how they do this, but none of them (have been shown to) do substantially better in the limit than a case analysis of all these possibilities.[27] The problem is that cases do not simply add up, they *multiply*: with n independent binary choices, there are 2^n cases to consider, too large for all but very small n.

This suggests one way to keep reasoning tractable: arrange the task so that the number of cases that have to be (conceptually) considered is kept manageable.[28] In what follows, I will present five different ways of doing just this. Each of them only deals with part of the problem and is limited in one way or another, but together they illustrate a range of approaches to reasoning that is sensitive both to the logic of what is believed and to the tractability of managing those beliefs.

7. VIVID KNOWLEDGE

The first way of keeping beliefs tractable is something I have discussed in (Levesque, 1986a). The idea is to require a KB to be in a certain syntactic form which I call *vivid* and which, for sentences of the predicate calculus, is this: a collection of ground, function-free atomic sentences, inequalities between all distinct constants (the so-called unique name assumption), universally qualified sentences expressing closed world assumptions (Reiter, 1978) over the domain and over each predicate,[29] and the axioms of equality. For example, the set

$$\{P(a), P(b), Q(a), R(b, c), (a \neq b), (a \neq c), (c \neq b),$$
$$\forall x[(x = a) \vee (x = b) \vee (x = c)], \forall x[Q(x) \supset (x = a)],$$
$$\forall x[P(x) \supset (x = a) \vee (x = b)],$$
$$\forall x \forall y[R(x, y) \supset (x = b) \wedge (y = c)]\}$$

together with the axioms of equality would constitute a vivid set of sentences.

The question of entailment for a collection of sentences of this form is considerably simplified. Letting KB $\models \alpha$ mean that KB entails α, then a KB that is vivid in form is first of all consistent and complete, that is, for every α in the language under consideration, either KB $\models \alpha$ or KB $\models \neg\alpha$ (but not both). Thus, for example, KB $\models (\alpha \vee \beta)$ iff

KB $\models \alpha$ or KB $\models \beta$, something that clearly does not hold in general for a KB that is incomplete. Similarly, KB $\models \exists x\alpha$ iff KB $\models \alpha^x_c$ for some constant c. Analogous considerations apply to the other logical connectives. Ultimately, entailment reduces to ground function-free atomic sentences. But in addition to being complete, a KB that is vivid in form also has the property that it entails an atomic sentence iff that sentence is one of its members. So entailment ultimately reduces to table lookup on atomic sentences. This property can be captured more formally as follows:

THEOREM 1. *Suppose KB is vivid and uses m constants. Let* $Q_1, \ldots, Q_n$ *be quantifiers, let* α *be quantifier-free. Then determining if KB* $\models Q_1 \cdots Q_n\alpha$ *has an* $O(m^{n+1}|\alpha|)$ *algorithm.*

Of course, this is exponential in n, but this is a situation where we do not want to be overly pessimistic in assessing worst cases. We are quite justified in assuming that n (actually, the depth of alternating quantifiers) is very much less than the size of α, which in turn is very much less than the size of KB, everything believed.[30]

Is this fast enough to be cognitively realistic? Not immediately, if $n = 3$, for example. But the proof of the theorem is based on the most naïve possible algorithm (that is, eliminate the logical connectives, and the quantifiers using the closed-world and unique-name assumptions, and do a linear search for atomic sentences), and we can do much, much better. By restricting a KB to be vivid, we have reduced entailment to *database retrieval* and so can expect to be able to handle very large KBs (for example, 10^9 atomic sentences, and with parallelism and further refinements, perhaps more).

But why should we care about a KB that is in this form? First of all, it is at the root of all the other approaches to tractable reasoning discussed below. But more directly and as discussed in (Levesque, 1986a), the atomic sentences of a vivid KB have a very important property: any model (in the logical sense) of the KB will have the same structure as that set of atomic sentences. In other words, there is a close correspondence between the syntactic form of the information and what the information is about: for each entity in the world it talks about, it has a constant symbol; for each group of entities that it claims stand in a certain primitive relationship, it has the corresponding constants syntactically related by appearing together in an atomic sentence of the appropriate kind; for each group of entities that it claims stand in a composite relationship, the corresponding constants also stand in a composite syntactic relationship. Unlike other logically complete collections of sentences, a vivid KB (or at least the atomic part of it) looks like its described subject matter. Thus we can think of a vivid KB as an *analogue* of the world it describes. If, for example, we want to find out how many entities have a certain property, we can answer by counting how many constants have the corresponding syntactic property in the KB. Similarly, if we want to represent a change in the world, we can do so by an analogous change in the KB itself. So we get a lot of the properties of mental models (Johnson-Laird, 1983) without having to postulate a non-sentential representation. This is a very powerful device since it allows us to reason about the world by operating *directly* on the symbolic structures.[31] In terms of case analyses, there is only a single case to consider, and it is determined explicitly by the form of the KB itself. This feature is exploited to great advantage in, among many others, research on qualitative simulation (de Kleer and Brown, 1983) where, for example, knowledge about a device is represented in vivid form, and the operation of the device is simulated by analogous operations on this representation.

8. REPRESENTING UNIVERSALS

Of course not everything we believe about the world fits into this form. We do appear to know a lot of universal facts (or rules) about what is around us. The second kind of KB to consider is what I will call a *Horn* KB: roughly, it consists of a vivid one (with suitably modified closed-world assumptions) and a collection of sentences (called Horn clauses) of the form

$$\forall x_1 \ldots \forall x_n((p_1 \wedge \ldots \wedge p_k) \supset p_{k+1}) \; n,k \geqslant 0, \; p_i \,\text{atomic.}$$

expressing general facts. Although this KB is not vivid in form, the information that it expresses is still complete, and so complex entailments can once again be reduced to atomic ones. These, however, cannot be calculated by table lookup. There is nonetheless the following result about non-quantificational Horn KBs (Dowling and Gallier, 1984):

THEOREM 2. *Suppose that* α *and the Horn clauses of a* KB *in Horn form are variable-free. Then* KB $\models \alpha$ *has an* $O(|\mathrm{KB}| \cdot |a|)$ *algorithm.*

Note that under these conditions, the entailment would be NP-hard if the KB was not in Horn form. We can also handle quantifiers provided that we do not use function symbols:[32]

THEOREM 3. *Suppose that* KB *is in Horn form and that* KB *and* α *are function-free. Then* KB $\models \alpha$ *has an* $O(m^{n+k+1}|\alpha|)$ *algorithm, where* m *is the number of constants in the* KB, n *is the maximum depth of quantifiers in* α, *and* k *is the highest arity of predicate being considered.*

As in the previous case, we can (and must) do much better than what these theorems suggest.[33] The main thing to notice about this generalization of vivid knowledge is that because of the Horn clauses, we are in the domain of *production systems* (Jackson, 1986). Much of the information required by the current generation of expert systems fits into this form: IF–THEN rules without recursion (or function symbols). So although still restricted in many ways, we are well on our way to covering what has been found to be useful in actually building knowledge-based systems.

9. SUBSUMABLE DISJUNCTION

So far we have stayed quite clear of incomplete knowledge. But there are going to be times when we simply must deal with disjunctive information, for example, one way or another. Consider the following two sentences:

Joe is 71 or 72 years old.
Joe is 71 years old or Mary has the flu.

The second sentence is a bit strange: nobody is going to tell you something like this under normal circumstances (logic puzzles being somewhat abnormal). The first sentence, on the other hand, has the kind of disjunction that appears to occur naturally, where there is some relationship between the two disjuncts.

It seems intuitively plausible that using a sentence like the first in conjunction with an existing KB should be easier to do than using the second. Imagine, for example, trying to figure out if Joe goes to kindergarten. With the first sentence, we should be able to avoid doing a case analysis: we can treat the first sentence as just a way of saying that Joe is in his early seventies (or is a senior citizen, or is old, or whatever), and combine it *as such* with the rest of the KB. With the second sentence, we do not have this option, and there is no alternative but to split cases when combining it with everything else we know.

To make this intuition more precise, call a KB in *semi-Horn* form if it can be divided into three parts as follows: a Horn KB that does not use some set of atomic formulas, a collection of disjunctions over those unused atomic formulas, and last, for each such disjunction, say $(p \vee q)$, a sentence of the form $(p \supset r) \wedge (q \supset r)$, where r is atomic. Intuitively, what we have is a Horn KB as before, some disjunctive information such that we know nothing specific about the individual disjuncts *except* what is provided by the third part of the KB, which gives us a *subsuming case* for the disjuncts. For example, p might say that Joe is 71 years old, q that Joe is 72, and r that Joe is in his early seventies, where absolutely nothing depends on Joe being exactly 71 or exactly 72. Once the precise details are pinned down, it is not too hard to show the following:

THEOREM 4. *Semi-Horn is no harder to deal with than Horn.*

There is simply no need to do a case analysis if nothing hinges on the specific cases. We can also go well beyond this. For example, we can consider a KB where the third part is a *taxonomy* of some sort that defines a hole range of subsuming cases (Brachman, 1983), and we can also consider using some of the disjuncts within the Horn KB in restricted ways. As long as we avoid multiplying the cases that need to be analysed, disjunction need not present a problem.

This technique of finding subsuming cases seems to handle much (almost all?) of the disjunctive information that occurs naturally. For example, uncertainty that arises from sensors is expected to lead to beliefs like *The value is 59, give or take 5%*, but not *The value is 14, 59 or 173*. We can normally handle this form of uncertainty with a subsuming *approximation*. Similarly, a plan to visit Chicago either March 3rd or 6th, can at first be blocked out as a plan to go in early March, and then resolved later.[34] Whenever cases are sufficiently close, we can often find a subsuming case, and postpone dealing with individual cases until we have enough information to settle on one. The overall effect is to make what is believed more vague, but more vivid as well.[35]

10. LIMITED INFERENCE

The next approach I want to consider is where we really do have to deal with cases that cannot be subsumed. Such situations do arise periodically (such as in logic puzzles), and the question is what to do about them.

Consider Figure 4, adapted from an example of Bob Moore: there are five blocks in a

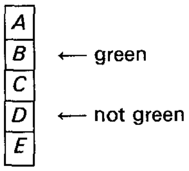

Fig. 4.

stack, where the second from the top is green, and the fourth is not green. Not a very complex situation. But notice that if I ask if there is a green block directly on top of a non-green one, the answer is not immediately obvious, even if you take everything you know about blocks and colours into account. If you think about it, though, there has to be one: if *C* is green, then because it is on *D*, it is the block in question; if *C* is not green, then because *B* is on it, *B* is the block in question. If we represent this situation by the KB

$$\{\text{On}(a, b), \text{On}(b, c), \text{On}(c, d), \text{On}(d, e), \text{Green}(b), \neg\text{Green}(d)\}$$

then sure enough, we have that

$$\text{KB} \models \exists x \exists y (\text{On}(x, y) \wedge \text{Green}(x) \wedge \neg\text{Green}(y)).$$

The point of this example is that perhaps it is too much to expect of our systems to be able to automatically determine the answer to this question, just on the basis of having the given sentences. People do not easily see the answer either. In other words, understanding the situation (as given by the sentences or the figure) should be different from knowing all of its properties.

What this suggests is that a knowledge representation and reasoning component needs to distinguish between what is explicit or evident in the beliefs, and what is implicit or can be inferred from the beliefs, given enough time and motivation. Ordinary logic is not going to make this distinction for us, but it is possible to take some steps in this direction, as I did in (Levesque, 1984), using a logic of belief.

One thing to observe is that a good deal of reasoning is based on detecting that a sentence and its negation are contradictory. With resolution, for instance, we start with $(p \vee q)$ and $(\neg q \vee r)$, and the reasoning is as follows:

> One of p or q is true and one of $\neg q$ or r is true; of the four possible cases, it can't be q and $\neg q$ *because q and $\neg q$ are contradictory*, and so it must one of the other three; in all three remaining cases, one of p or r is true, and so $(p \vee r)$ must be true.

Now consider a shallow reading of sentences where the contradiction between a sentence and its negation is *not* observed. In this setting, the reasoning would not go through. More precisely, a *surface interpretation* of a sentence is like a normal logical interpretation (defined recursively in the usual way) except that atomic sentences and their negations can be given independent truth values.[36] For example, a surface reading might (mistakenly) take both Green(c) and

$\neg$Green(c) to be true, or neither. Just as KB $\models \alpha$ is defined as holding iff all logical interpretations satisfying KB also satisfy α, define KB $\Rightarrow \alpha$ to hold iff all *surface* interpretations satisfying KB also satisfy α.[37] The suggestion here is that this second form of entailment should be considered as a reasoning task.

The result demonstrated in (Levesque, 1984) is the following:

THEOREM 5. *Suppose* KB *and* α *are arbitrary non-quantificational sentences in conjunctive normal form. Then*

(*a*) *determining if* KB $\models \alpha$ *is co-NP-hard* (*that is, likely intractable*), *but*

(*b*) *determining if* KB $\Rightarrow \alpha$ *has an* $O(|KB| \cdot |\alpha|)$ *algorithm.*

Thus, $\Rightarrow$ is a provably tractable subset of $\models$. It also is an independently motivated one:

THEOREM 6. KB $\Rightarrow \alpha$ *iff* KB *tautologically entails* α *in the system of Relevance Logic of* (*Anderson and Belnap* 1975; *Dunn* 1976).

This does not completely solve the problem, however. For one thing, the obvious generalization of this definition (and of tautological entailment) to deal with quantifiers is undecidable.[38] Another thing is that this form of limited reasoning (and of tautological entailment) has quirks. For example, although α and $(\neg\alpha \vee \beta)$ do not tautologically entail β, they do tautologically entail $(\beta \vee (\alpha \wedge \neg\alpha))$, which may (or may not) be justifiable in terms of Relevance Logic, but looks mighty suspicious in terms of explicit belief.

How to get around these defects remains a very open question. Nonetheless, the approach is quite attractive because it is completely general purpose: it does not require the (form of the) KB language to be restricted in any way.[39] As such, it can be thought of as the quick surface reasoning that is always (reliably) done prior to any form of deep logical analysis or problem solving (and see the discussion of puzzle mode below).

11. CASE ELIMINATION

The reasoning we have considered up to now has always been logically sound, and until the previous section, logically complete as well. In this section, we will consider the application of a type of logically unsound reasoning, known as *non-monotonic reasoning* (Reiter, 1987).

Consider the situation depicted in Figure 5: there are three cages stacked up containing

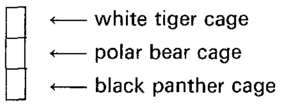

Fig. 5.

(from the top to bottom) a white tiger, a polar bear, and a black (that is a non-white) panther. The question here is whether there is a white animal directly above a non-white one.

This is really just a disguised version of the puzzle from the previous section. But now the question seems easier to answer. If we are not barred from using what we know about animals and colours, we are tempted to reply: "Sure. It's the polar bear." Put another way, there is an overwhelming tendency to believe that a polar bear is white, if not contradicted by the known facts, and from there, the answer is immediate.

But as in the previous example with the blocks, no assumption about colours is really necessary: the answer would be yes even if the polar bear had been spray-painted green. What the assumption does is allow us to answer the question without even considering the non-white case, like we had to do with the blocks. Thus the default assumption has the effect of making the KB more vivid, by deciding the previously unspecified colour of the polar bear. From there, reasoning is again tractable, although there will be conclusions that are *not* logically implied by the original beliefs.

So given appropriate default information for a wide variety of kinds and properties, it should be possible to make a KB much more vivid, to significant computational advantage. Moreover, if the defaults are chosen properly (and, for example, reflect statistical occurrences) reasoning errors will be minimized.[40] Moreover, this seems to be in accordance with the way people do imagine the given situation (that is, it is hard to think about a polar bear without thinking of it as being white).

There are three main issues to be addressed. First, we need a language to represent the default information so that it can be incorporated within a KB. Ordinary first-order logic allows us to say

All polar bears are white.

but not the more accurate

Polar bears are white.

that admits exceptions (Carlson, 1982). Secondly, we must determine what it means to reason *correctly* with these defaults. Reasoning with assumptions is logically unsound, but it is not unsystematic. These two issues have occupied most of the considerable body of research in this area (Reiter, 1987). But a third and very important issue for our approach is to determine how difficult it will be to actually apply the defaults. If producing a vivid KB using defaults is itself intractable, we are no further along in explaining the role of these defaults in cognition.[41] Fortunately, some positive results are beginning to appear in this area (Selman and Kautz, 1988).

12. ODDS AND ENDS

In this section, I will briefly go over three more approaches to the tractability issue that are somewhat less developed (to put it mildly).

First of all, picking up from the previous section, we should consider making a KB vivid even if there are no appropriate defaults to use. Note, for example, that it is difficult to think about a dog chasing a cat around a tree without also thinking about it as a clockwise or counterclockwise motion, even though there is no real default direction. Nonetheless, assumptions about all sorts of observer-centered visually salient properties (that is, features that someone in the imagined situation could not help but observe) are irresistible.[42] Another avenue is to eliminate universals in favor of a vivid KB that has a number of individuals satisfying the universal, along the lines suggested in (Johnson-Laird, 1983).

Another very different approach (that ties in with the comments below on puzzle mode) is the notion of meta-reasoning. One way to consider controlling and limiting otherwise unrestricted reasoning is to trap at decision points, shift up a level, and meta-reason about how to proceed (or whether to quit) (Genesereth, 1983; Kramer, 1984; Laird *et al.*, 1987). This is especially important in cases where reasoning tasks can fall into patterns for which there are known techniques. An example is the realization that a reasoning task reduces to solving a set of linear equations which can be handled by Gaussian elimination (and there are many such special-purpose efficient reasoners). Of course meta-reasoning can lead to meta-meta-reasoning and so on. What is needed here is a form of *convergence* where the meta-reasoning task is always *easier* than the original one, so that at some level, the reasoning can be handled without further introspection. Although thinking produces thought, ultimately, it should not *require* thought. In this sense, it is not so different from low-level perception or motor control.

Another approach worth considering (though less motivated by psychological concerns) is limited theorem proving. Normal theorem provers (like resolution) are guaranteed to be logically correct, but are not guaranteed to terminate quickly. We should also consider theorem-proving algorithms where the opposite is true (that is, algorithms that are designed to run in polynomial time, as opposed to time-limited versions of otherwise exponential algorithms). One such algorithm (that I have been exploring) is the following (roughly): to find a satisfying interpretation for a set of propositional clauses, choose a short clause whose atomic sentences appear most often: then, for each assignment to these atomic sentences, simplify the set of clauses under this assignment and iterate the entire procedure with the resulting set of clauses that looks the most likely to be satisfiable. Assuming the estimate of satisfiability can be done quickly, the entire "local" theorem-proving procedure is guaranteed to terminate quickly. Obviously, the real issue here is the set of inputs for which the procedure can be guaranteed to work correctly. But failing this, preliminary experimental evidence seems to indicate that it does remarkably well even on apparently difficult cases (sets of clauses that are uniquely satisfiable).

13. PUZZLE MODE

The constraints of physical realizability have pushed us towards a computational architecture that (in terms of reasoning) avoids unrestricted inference over the full language of first-order logic. But is this right? People are, after all, capable of doing full logical inference given some training, a lot of time, perhaps some memory aids, coffee, and a heap of motivation. How could a limited reasoning architecture ever do unlimited reasoning? To resolve this paradox, consider the following two analogies.

First, a finite state transducer.[43] As a computational architecture, it is very limited: it cannot do multiplication, and it can only do addition when numerals are presented to it least significant digit first. Now imagine a machine that is like a finite state transducer except that whenever it emits a certain character, say a $*$, the subsequent inputs it receives are the outputs it produced since the last time it emitted a $*$. If we now ask what these machines can do, the answer is *everything*: they are equivalent to Turing machines. To see this, note that they can start by producing as output a description (in some form) of the initial configuration of a Turing machine, followed by a $*$; subsequently they need only be able to generate the next configuration of the Turing machine given the previous one, something that a finite state transducer *can* do. Although the resulting system is no longer a simple finite-state transducer, there is another way to look at this: we might say that the machine itself has not really changed, but the *environment* it runs in now cooperates by providing it with a memory buffer. The key point though is that instead of trying to do *arithmetic*, the machine now does *a Turing machine doing arithmetic*, that is, it is one meta-level removed from its initial task.

As a second example, consider the PROLOG programming language. It is based on a form of resolution called SLD-resolution (Kowalski and Kuehner, 1971) that is complete for Horn clauses, but not for general clauses. That is, the PROLOG processor is not a complete theorem prover for full first-order logic. But PROLOG is a programming language, and it is possible to write in PROLOG a resolution theorem prover. The resulting system *is* a complete theorem prover. Again, the PROLOG processor has not really changed, but instead of running over the clauses directly, it runs over a theorem-prover. As above, instead of trying to do *full theorem proving*, the machine now does *a resolution theorem prover doing full theorem proving*, and again, it is one meta-level removed from its initial task.

I believe that these two examples are the way to think about people doing full logical reasoning. It is not that we have to consider augmenting the computational architecture, but we have to be able to go into a puzzle mode that is one meta-level up from the original task. In other words, instead of being able to reason in a very advanced way, the architecture need only be able to reason about *the steps to follow in reasoning in this advanced way*. The expectation is that (as in the above two examples) this can be done with modest architectural means.[44]

If this seems implausible, consider finding a square root using Newton's method. In one sense, we do not want to claim that people have an architecture that is capable of using this method, since before Newton, nobody could. And yet people can certainly do so once they are taught the method. The way to understand this, I think, is to say that we are born with an architecture that allows us (given appropriate memory aids and the rest) to follow a step-by-step procedure, and that we can learn various such procedures.

This is certainly compatible with the logic case. People who have not been trained in logic cannot do full logical reasoning, although as I have emphasized thoughout, much of what they do is indeed based on logic. From this point of view, the empirical question is just how much reasoning (logical or otherwise) has to be acquired as a skill in this way (as a method to be used in certain circumstances), and how much is architecturally available. The trouble is that the transition from puzzle level to object level might be very smooth, and introspection is a notoriously unreliable guide.[45]

14. CONCLUSION

To conclude, let me simply summarize the main points without further comment: Contrary perhaps to common folklore, logic does have a significant role to play in computational approaches to the study of cognition, but it must be tempered by concerns of computational complexity. Cognitive models, computational or not, need to be physically realistic. But this physical realism need not be studied solely at the level of mechanism (algorithms and data structures, or even biology); it can be approached at the level of computational tasks. The thesis is that when inputs cannot be bounded in size, a computational task outside of P is intractable. Knowledge representation and reasoning, as it has been presented here, can be thought of as the investigation of computationally tractable and semantically coherent manipulations of large collections of sentence-like symbolic structures. Classical logic is the base camp for such an investigation, but there are large, virtually unexplored areas nearby. And finally, settling in these areas is perhaps the only rational way a creature has of dealing with a complex informational environment.

But what does all this prove? Not a whole lot yet, I should say. But like a lot of research in artificial intelligence, it does at least suggest that there is room in cognitive science for a computational study of how we *differ* from the other animals. And that, these days, is something.

NOTES

* This paper is based on a seminar I gave in 1987 at York University, Carnegie Mellon University, Harvard University, the University of Western Ontario, and the University of California at San Diego. Thanks to my hosts at these institutions, and to all the others who attended the lecture and tried to keep me honest. Thanks also to Wolfgang Bibel, Ron Brachman, Tim Horton, David Israel, Joe Nunes, Ray Reiter, Jim des Rivières, Bart Selman, Rich Thomason, and John Tsotsos for reading an earlier draft and snaring several falsehoods. The ones that got away are not their fault, of course. Financial support was gratefully received from the Natural Sciences and Engineering Research Council of Canada.

[1] In what follows, I will assume some familiarity with the basic concepts of (sequential) complexity theory, namely the idea of a lower bound, the classes P, NP, and the properties NP-complete and NP-hard. Precise definitions for all of these can be found in (Garey and Johnson, 1979). For a nicely written account of computation and complexity for the non-specialist, see (Harel, 1986).

[2] There have been some exceptions, however, most notably Christopher Cherniak (Cherniak, 1986), and outside the area of pure cognition, (Tsotsos, 1988) in vision, and (Barton *et al.*, 1987) in natural languages, among others.

[3] The computational viewpoint also shows some of the limitations of the other two vantage points: imagine trying to interpret the activity of a personal computer (that is calculating a tax return, say) either in terms of the behaviour of individual transistors, or in terms of the statistical properties of collections of transistors.

[4] Following common practice in computer science, I will use the term "symbol" to mean a pattern of relatively passive component states that the more active components of a computational architecture can recognize, differentiate from others, and manipulate. The usual examples are strings of bits, letters, or digits. The symbols may or may not *represent* anything, so maybe the word "token" or "character" would be more appropriate.
[5] Obviously, the sensors and effectors must satisfy a number of laws too, but I will not discuss these further. For better or for worse, computer science has focussed by and large on the central box in the picture. I will follow this practice and assume that the sensors and effectors (such as keyboards, laser printers, television cameras, robot arms and the like) are best described in other than computational terms.
[6] I'm not trying to make a very deep point here about formality or anything else. This is just entry level computer science: what you have to tell first-time programmers about why a recipe is not a program, or how programming is different from giving directions to a friend.
[7] It is perhaps even more surprising, given this emphasis on symbol manipulation, that operating systems, robot manipulators, real-time controllers, and other systems that directly interact with their surrounding environment can also be programmed as computational systems in our sense. Indeed, the discovery that multiple asynchronous processes can be successfully managed in terms of general symbolic operations, that is, using software techniques (like monitors) instead of hardware techniques (like interrupts), is a fairly recent and significant development in our understanding of programming.
[8] In what follows I will treat *time* as the single important resource to consider, although a story ultimately needs to be told about other resources too (memory, processors, *etc.*).
[9] For example, lifting some figures from (Cherniak, 1986), suppose that the fastest computational system requires 2^n sequential steps for an input of size n (by some measure). Then even if the fastest step can be done in the time it takes light to cross a proton (10^{-23} seconds), it would still take longer than the age of the universe to complete the task for any $n > 137$.
[10] What I do preclude in this second approach would be a computational architecture that has *no* task, that is not seen as computing outputs given inputs (for example, a machine that simply cycles forever through some internal states never producing output of any sort).
[11] In many cases, and especially for the knowledge-based systems discussed below, instead of thinking in terms of input and output streams, we think of inputs as *accumulating* internally over time, with the understanding than an output may be arbitrarily far removed from an input that it depends on. In this case, the size of the intervening stream of inputs must be taken into account.
[12] See (Tsotsos, 1988) for an analysis of low-level vision along these lines.
[13] This is not to say that computer scientists do not study models of computation where this thesis is false. Examples are machines that can grow an exponentially large number of processors and machines with oracles of various kinds. The claim is just that such machines cannot be *built*, that is, physically realized in a way that would secure a greater than polynomial speedup over ordinary sequential Turing machines.
[14] Very strictly speaking, this would not be true of someone sufficiently confused about crocodiles and steeplechases to think that this question required tackling an even more difficult problem, like the halting problem for a Turing machine. But the less said about such cases, the better.

[15] There is, however, a brand of artificial intelligence research whose goal is to produce working systems by whatever method. In this case, the computational artifact does indeed have intrinsic value, independent of what it tells us about any cognitive activity.
[16] Except for intentionally behaving philosophically, these decisions are not made in puzzle mode, up a recursive ladder of indecision.
[17] This is not to suggest that we have control over *all* of our cognitive resources. Sensory acuity and short-term memory capacity, for example, are both limited in ways that are apparently beyond our control.
[18] If the only problematic cases are ones that do not seem to occur in practice (for example, if they require inputs that are too large), we can simply decide to eliminate them from consideration (without great loss of generality, presumably). But this does not eliminate our concern with extreme cases; it merely changes what cases we consider to be extreme.
[19] A potential counter-example to this claim was the task of linear programming: the simplex algorithm for linear programming is one that is worst-case exponential but works superbly in practice. As it turned out, the task itself was discovered to be in **P** after all, and polynomial algorithms comparable to simplex are under development. See (Megiddo, 1987) for a review of this area.
[20] Following the usual AI custom, I will use "knowledge" and "belief" interchangeably to mean belief.
[21] In what follows, I will use (English) sentences to talk about these symbolic structures, but of course, they need not have this syntactic form at all.
[22] The perception and action components also perform a type of reasoning, of course, and will generate hypotheses and plans that need to be managed as well.
[23] Of course, there is a lot more to *thinking* than dealing with perception and action, or for that matter, reasoning of whatever kind. Regrettably, I have nothing more to say about this other type of thinking. Nor will I have anything to add on perception, induction, learning, planning, explanation, and any number of other forms of reasoning. The relevance of logic to all of these is something that I think requires separate analyses. See also (Levesque, 1987) and subsequent articles in the same issue for contrasting opinions on this.
[24] Connectionists, for example, apparently do not.
[25] Imagine an expert analyzing her own behaviour: "Did I do so-and-so? Well, it *must* have been because I thought that such-and-such."
[26] This is ground I have covered elsewhere. See (Levesque, 1986b).
[27] This is not to say that all proof methods are computationally equivalent. Resolution theorem proving, for example, seems to do much better in practice than methods based on truth-tables. In the quantificational case, the general treatment of variables allowed by unification is a significant advantage. However, the problem, nonetheless, is that finding a resolution proof (even in the non-quantificational case) may require an exponential amount of work, and worse, it was recently shown that even if we could zero in on a proof quickly, there are cases where the shortest resolution proof itself may be exponentially long (Haken, 1985).
[28] It is not sufficient to know that there is only a single case to consider, if you do not know what that case is. For example, finding a satisfying interpretation for a uniquely satisfiable set of clauses is apparently intractable (Valiant and Vazirani, 1986).

[29] Strictly speaking, Reiter's version of the closed world assumption consists of the negation of every atomic sentence not entailed by the KB, and not the quantified sentences I am using.
[30] For all but mathematical examples, an assumption that $n \leqslant 3$ seems plausible, and exceptions to this may require a different approach (see Section 13).
[31] Contrast, for example, doing arithmetic directly over standard number representations with reasoning about the answer using Peano's axioms.
[32] The proof of this result simply adapts the previous theorem given a finite Herbrand universe.
[33] We can handle function symbols as well, if we can control the depth of self-nesting.
[34] Indeed, if this strategy does *not* work during planning, we really do have a hard time dealing with the exploding contingencies, and insist on resolving them as soon as we can.
[35] We can contrast vagueness here with *ambiguity*: only with the latter are we committed to resolving the uncertainty to a single specific case.
[36] Surface interpretations thus form a proper superset of the ordinary logical interpretations.
[37] If KB $\Rightarrow \alpha$ then KB $\models \alpha$, but not vice-versa. That is, the inferences sanctioned by $\Rightarrow$ are logically sound, but not logically complete.
[38] This defect can be remedied, however, as demonstrated in the work of Patel-Schneider (Patel-Schneider, 1985).
[39] The conjunctive normal form requirement should not present a problem. One would expect to build up a KB conjunctively, where each new addition is small compared to the size of the entire KB, and so can be converted relatively easily to conjunctive normal form, and appended to the KB.
[40] A true cost-benefit analysis of default reasoning should also take into account the *seriousness* of errors, for example, the difficulty of recovering from them.
[41] This issue will of course influence how we resolve the first two.
[42] In (Levesque, 1986a), I suggest that we have learned to depend so much on visual information precisely because of what it cannot leave unsaid about the observed situation (compared to unrestricted linguistic information), that is, its vividness and the tractability this guarantees.
[43] This example was derived jointly with Jim des Rivières.
[44] This is not unlike Simon's ant (Simon, 1969), where rich behaviour results from a weak mechanism in a rich environment.
[45] Although I have absolutely no hard evidence for this, I believe that *reasoning by cases* (or any reasoning that does not follow a single line of thought) is something that is only done in puzzle mode when you say to yourself "Well, let's see. There are only four possibilities . . ." I also suspect that people are the only animals with such capabilities. Here, then, is a token falsifiable prediction: we will never discover evidence of a non-human species behaving in a way that can *only* be explained as reasoning by cases.

REFERENCES

Anderson, A. and Belnap, N.: 1975, *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, Princeton, NJ.

Ballard, D.: 1986. Cortical connections and parallel processing: structure and function. *The Behavioral and Brain Sciences* **9**(1) 67–90.

Barton, G. E., Berwick, R., and Ristad, E.: 1987, *Computational Complexity and Natural Language*. MIT Press, Cambridge, MA.

Brachman, R.: 1983, What is-a is and isn't: an analysis of taxonomic links in semantic networks. *IEEE Computer* **16**(10): 30–36.

Brachman, R. and Levesque, H.: 1984, The tractability of subsumption in frame-based description languages. In *AAAI-84*, pages 34–37, Austin, TX.

Carlson, G.: 1982, Generic terms and generic sentences. *Journal of Philosophical Logic* **11**: 145–182.

Cherniak, C.: 1986, *Minimal Rationality*. Bradford Books, Cambridge, MA.

Church, A.: 1936, A note on the Entscheidungsproblem. *The Journal of Symbolic Logic* **1**: 40–41.

Cooks, S.: 1971, The complexity of theorem proving procedures. In *The 3rd. Annual Symposium on the Theory of Computing*, pages 151–158, New York, NY.

de Kleer, J. and Brown, J. S.: 1983, The origin, form, and logic of qualitative physical laws. In *IJCAI-83*, pages 1158–1169, Karlsruhe, West Germany.

Dowling, W. and Gallier, J.: 1984, Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming* **3**: 267–284.

Dreyfus, H. and Dreyfus, S.: 1986, *Mind over Machines*. Macmillan, New York, NY.

Dunn, M.: 1976, Intuitive semantics for first-degree entailments and 'coupled trees'. *Philosophical Studies* **29**: 149–168.

Garey, M. and Johnson, D.: 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA.

Genesereth, M.: 1983, An overview of meta-level architecture. In *AAAI-83*, pages 119–124, Washington, D.C.

Haken, A.: 1985, The intractability of resolution. *Theoretical Computer Science* **39**: 297–308.

Harel, D.: 1986, *Algorithmics: The Spirit of Computing*. Addison-Wesley, Don Mills, Ontario.

Hayes, P.: 1987, A critique of pure treason. *Computational Intelligence* **3**(3): 179–185.

Jackson, P.: 1986, *Introduction to Expert Systems*. Addison-Wesley, Don Mills, Ontario.

Johnson-Laird, P.: 1983, *Mental Models*. Harvard University Press, Cambridge, MA.

Kirousis, L. and Papadimitriou, C.: 1985, The complexity of recognizing polyhedral scenes. In *The 26th Annual Symposium on the Foundations of Computer Science*, Portland, Oregon.

Kowalski, R. and Kuehner, D.: 1971, Linear resolution with selection functions. *Artificial Intelligence* **2**: 227–260.

Kramer, B.: 1984, Representing control strategies using reflection. In *CSCSI-84*, pages 153–158, London, Ontario.

Laird, J. E., Newell, A., and Rosenbloom, P. S.: 1987, Soar: an architecture for general intelligence. *Artificial Intelligence* **33**(1): 1–64.

Levesque, H.: 1984, A logic of implicit and explicit belief. In *AAAI-84*, pages 198–202, Austin, TX.

Levesque, H.: 1986a, Making believers out of computers. *Artificial Intelligence* **30**(1): 81–108.

Levesque, H.: 1986b, Knowledge representation and reasoning. *Annual Reviews of Computer Science* **1**: 255–287.

Levesque, H.: 1987, Taking issue: guest editor's introduction. *Computational Intelligence* **3**(3): 149–150.

Megiddo, N.: 1987, Linear programming (1986). *Annual Reviews of Computer Science* **2**: 119–145.

Moore, R.: 1982, The role of logic in knowledge representation and commonsense reasoning. In *AAAI-82*, pages 428–433, Pittsburgh, PA.

Newell, A.: 1982, The knowledge level. *Artificial Intelligence* **18**(1): 87–127.

Patel-Schneider, P.: 1985, A decidable first-order logic for knowledge representation. In *IJCAI-85*, pages 455–458, Los Angeles, CA.

Pylyshyn, Z.: 1984, *Computation and Cognition: Towards a Foundation for Cognitive Science*. Bradford Books, Cambridge, MA.

Reiter, R.: 1978, On closed world databases. In Gallaire, H. and Minker, J., editors, *Logic and Databases*, pages 55–76, Plenum Press, New York, NY.

Reiter, R.: 1987, Nonmonotonic reasoning. *Annual Reviews of Computer Science* **2**: 147–186.

Rumelhart, D. and McClelland, J. (editors): 1986, *Parallel Distributed Processing*. MIT Press, Cambridge, MA.

Selman, B. and Kautz, H.: 1988, The complexity of model-preference default theories. In *CSCSI-88*, Edmonton, Alberta.

Simon, H.: 1969, *The Sciences of the Artificial*. MIT Press, Cambridge, MA.

Tsotsos, J.: 1988, A complexity level analysis of immediate vision. *International Journal of Computer Vision* **4**: 303–320.

Valiant, L. and Vazirani, V.: 1986, NP is as easy as detecting unique solutions. *Theoretical Computer Science* **47**: 85–93.

Department of Computer Science and
Canadian Institute for Advanced Research,
University of Toronto,
Toronto M5S 1A4,
Canada