Lucid Representations*

Ernest Davis
Courant Institute
New York, NY

May 25, 1994

Abstract

This paper criticizes the widespread idea that knowledge bases in AI systems should be complete and that representations should be "model-like." The arguments in favor of such representations are less cogent and more ambiguous than they appear at first. Levesque's suggestion that representations should be "vivid" is extremely restrictive, particularly in its uniform imposition of a closed-world assumption. Spatial representations that are adequate for reasoning about a wide range of physical phenomena must ultimately either use complex symbolic reasoning or deal with partial and imperfect approximations. Requiring that temporal representations be fully detailed simulations will often be extremely inefficient. Finally, a flexible intelligent system must necessarily deal with partial information of all kinds, and the techniques for carrying out reasoning about partial information using complete representations are very limited in their application.

There is a recurrent idea in the AI research community that representations should resemble models, rather than being an arbitrary collection of formulas or constraints. Like a model, a representation should correspond to one particular way the world could be, certain and complete. Like a model, the representation should characterize the world directly and perspicuously, not give a collection of interacting conditions on it. Answering questions from a model is a matter of checking and measuring rather than of reasoning; so it should be from a representation. This argument has appeared in a variety of forms and contexts. Levesque [1986, 1989] proposes that the core of a knowledge base should be a "vivid" representation, a collection of atomic formulas satisfying the closed world assumption. Halpern and Vardi [1991] propose that reasoning about knowledge should be performed by explicitly constructing Kripke systems of possible worlds. Forbus, Nielsen, and Faltings [1987] argue that spatial representations for spatial or physical reasoning programs should contain a "metric diagram", a representation that gives full geometric details. The vast majority of physical reasoning programs that do spatial reasoning do use such a representation (e.g. [Gelsey, 87] [Joskowicz, 87], [Faltings, 87], [Funt, 85] [Gardin and Metzler, 89]. Exceptions include [de Kleer, 77] and [Randell and Cohn, 89].) Miller [1985], as well as others, has suggested that planning programs should always use construct instantiated plans, to avoid the complexities involved in reasoning about partial plans. In this paper, I will generically call all such representations "lucid." ("Vivid" would be a better word, but Levesque has appropriated it for his own theory.) Representations that are not lucid are "indirect".

The aim of this paper is to argue that, despite their attractiveness, lucid representations are often inappropriate or inadequate for AI systems.[1] Though it is certainly reasonable to limit very

*This paper was largely inspired by the author's participation in the Workshop on Representations in Mental Models, Cambridge, Mass., March 12-13, 1990. I thank the participants in the workshop, particularly Whitman Richards, who organized it, and Pat Hayes and Drew McDermott, who have made many helpful suggestions and criticisms. The writing of this paper was supported by NSF grant # IRI-9001447

[1] Doyle and Patil [1991] make a similar argument for a definitional knowledge base. I shall be concerned exclusively with assertional knowledge bases.

substantially the range of partial knowledge expressed in a knowledge base, a complete exclusion of partial knowledge is not generally feasible. In section 1, I will consider some of the general arguments for lucid representations, and show that they are less cogent than they at first appear. Section 2 examines Levesque's proposal for vivid representations. Sections 3 and 4 consider lucid representations of spatial and temporal information. Section 5 argues that there are circumstances where indirect representations are more appropriate than lucid representations, even though complete information is in principle available. Section 6 argues that AI programs must deal with a wide range of partial information, and that it is very doubtful that reasoning with partial information can in general be carried through using lucid representations.

1 Motivations

There are a number of different motivations for preferring lucid to indirect representations. I will enumerate them in decreasing order of cogency, and then briefly contest them.

1. People often prefer to think in concrete rather than abstract terms. In reasoning, learning, communicating, and teaching, people invest considerable effort to construct mental devices such as diagrams, models, paradigmatic examples and counter-examples, parables, allegories, and precedents, in order to augment abstract principles with concrete instances.

2. As a general rule, in a given type of inference, strengthening the given information and weakening the desired conclusions makes the process of inference easier, just as it is easier to shop if your store is well stocked and your shopping list is short. Therefore, the inference process will be easiest if complete information is given.

3. Computation theory seems to prefer lucid representations. Computational complexity is often associated with partial knowledge. Generalizing a problem by adding disjunction, replacing constants by variables, replacing a total ordering by a partial ordering, or replacing exact values by constrained values, will often turn an easy problem into an intractable one.

4. It is almost always easier to devise algorithms for lucid representations. Most algorithms in the literature are designed for representations that are lucid in most respects.

5. Lucid representations are much easier to explain and to show. In particular, in dealing with geometric representations, a lucid representation can be *drawn*, which enormously facilitates explanation, coding, debugging, user interface, and so on. Constructing intelligible diagrams to illustrate the meaning of a indirect spatial representation is difficult to do by hand and very difficult to automate (see, for example, [Davis, 86]).

Arguments (4) and (5) are certainly important over the short run, for implementing a reasoning system, particularly a spatial reasoning system, over the next year or the next five years. However, they clearly have very little cogency as an argument for the ultimate direction of AI. No one ever said that AI should be easy.

The general rule invoked in argument (3), that partial knowledge is computationally difficult, is broadly true but not very useful in any particular case. The evaluation of which representation works best for a given cognitive task depends quite delicately on the specific details of the computational task, of the problem space, and of the representations, and there is no point in trying to prejudge the issue. In fact, as we shall show below, there are many cases where indirect representations leads to a considerable gain in computational effectiveness.

Argument (2) cuts both ways. Inference will be easier if complete information is given, but it will be harder if complete information is required. If an inference process that goes through several levels, reasoning from a knowledge base of type A to one of type B, and from type B to type C, then requiring that type B contain complete information will make the second inference easier but

the first more difficult. The latter effect may well dominate the former. Some examples will be presented in section 5 below.

Moreover, the knowledge base of an autonomous agent is not "given"; it is acquired, primarily from perception or natural language text. To require that this knowledge always be complete is to put a heavy burden — usually, indeed, an impossible burden — on the processes of interpreting perception and text. (It may be noted that there has been little interest in lucid representations in the natural language community, since the content of natural language texts are significantly incomplete more often than not.)

Finally, this argument addresses only the "amount" of information contained in the knowledge base, not the form in which it is expressed. As we shall see in section 5, there are cases where inference will be faster if the knowledge base contains facts that are indirect in form.

As for argument (1), this observation of human preferences is interesting and undoubtedly of great ultimate importance, but it is not easy to interpret. In particular, it does not actually suggest that the cognitive representations used by people are lucid in anything like the senses under discussion.[2] It is true that "people resort to diagrams and models for all but the simplest spatial problems" [Forbus et al., 87]. However, a diagram need not have accurate metric information to be useful. Many diagrams (e.g. circuit diagrams or subway maps) have only topological information; many diagrams have details enlarged for clarity; many diagrams are just crude sketches. Indeed, a schematic diagram of a complex physical system is often more useful than a detailed picture, even though the latter contains more information. Moreover, people find diagrams, such as flowcharts, useful even in domains where there is no spatial information and only a crude, and not very helpful, analogy to spatial topology. Why this should be so is a considerable mystery, though it is presumably based on the fact that people's visual apparatus is a particularly sophisticated part of their cognitive machinery [Sloman, 75]. (Levesque [1986] suggests, tentatively, that it may be that visual information is inherently more tractable than linguistic information. However, it is hard to believe that inferring that $A \cup (B \cap C) = (A \cap B) \cup (A \cap C)$ can be performed more efficiently by first generating and then interpreting a geometric Venn diagram than by using the laws of set theory.)

The same is true of the other cognitive tools mentioned above. It is true that an instructor who teaches purely in terms of abstract principles without giving examples is making a big mistake. But an effective example is not a fully detailed account; it is a partial account, crafted so as to emphasize the point being made as brightly as possible. Again, effective historical or biographical or journalistic writing is not a fully detailed chronological account. The details that are included and excluded must be chosen carefully, and the order of presentation is chosen according to very subtle principles.[3]

In any case, it is far from clear that the form in which it is easiest for humans to assimilate information bears much resemblance to their internal representations. One can just as plausibly make the following two arguments against the idea that people's representation of spatial representation is quite unlike a picture: (1) People have great difficulty drawing a realistic picture, or even a picture that satisfies themselves, even of familiar locales and objects, without very extensive training [Tversky, 81]. (2) The very fact that people do need diagrams for metric judgements suggests that they have difficulty generating mental diagrams. If they had a picture in their heads, why would they need the external picture? I don't say that either of the arguments is tremendously cogent, but they are at least as good as the reverse argument.

Lucid representations have an appeal that goes beyond the arguments above, based on the fact that they often seem easier than they actually are. There are a couple of reasons for this illusion.

[2] There is a body of psychological experimentation that has been interpreted as indicating that people use such lucid representations as occupancy arrays [Kosslyn, 80]. This argument has been pursued and debated at great length (see, for example, [Pylyshyn, 84] for the opposing views) and I am not able to add anything to this debate. In any case, even the adherents of these theories have not suggested how such representations could be used for the types of inferences I shall discuss below.

[3] The "cases" that form the basis for case-based reasoning are likewise partial rather than complete descriptions [Slade, 91]. Case-based reasoning therefore lies outside the scope of this critique.

First, there is what McDermott (1981) called the "wishful control structure problem:" "The idea that because *you* can see your way through a problem space, your program can." Since it tends to be relatively easy for people to see the consequences entailed by a lucid representation, they assume that it will be easy for the computer.

Second, since lucid representation are easily explained and understood, researchers often assume that they in fact are self-explanatory, and that defining their semantics formally is just a pointless exercise in formalism. However, in fact many of the properties of a lucid representation depend on subtleties in the way its semantics are defined. If the semantics are left vague, researchers can play an unconscious shell game, in which, whenever a limitation of the representation is revealed, the researcher can make a slight shift to the semantics or to the details of the representation to cover it. In this way, he can go a long time without realizing that it is not possible to cover all the limitations simultaneously. We will illustrate some examples of this shell game below. The situation is reminiscent of the state of semantic nets in the early 70's; since the meaning of a semantic net was self-explanatory, people were quite unconsciously using them in wholly incompatible ways [Woods, 75] [McDermott, 81].

Another form of shell game is to assume that, because a lucid representation is good for one kind of inference, it will necessarily be good for another. For instance, I have often seen the following argument presented: It is known [Dean and Boddy, 88] that determining the correctness of a non-linear plan with context-dependent actions is an NP-hard problem, while determining the correctness of a linear plan is linear time. Therefore, it is argued, planners should search through a place of linear plans rather than a space of non-linear plans. But the argument is a non-sequitur; the fact that verification is easier does not at all justify the conclusion that searching will be easier. In fact, in virtually any theory of planning, finding a correct plan is NP-hard. Whether in practice linear planning will be better than non-linear planning is an entirely open question. (The experimental studies of Soderland and Weld [1991] suggest that a complete non-linear planner is likely to be significantly more efficient than a complete linear planner.)

2 Levesque's vivid representations

Levesque's Computers and Thought lecture (1986) proposed that AI knowledge bases be, as far as possible, implemented as vivid representations. A vivid representation is very much like a relational database. Formally it may be defined as follows (this definition and example are from [Levesque, 86]):

Definition: A vivid knowledge base can be viewed as a first-order theory with equality containing

i. A collection of ground, function-free, atomic sentences (i.e. predicates applied to constants.)

ii. Inequalities between all distinct constants (the unique-names assumption.)

iii. The closed-world assumption on entities: every entity is named by a constant symbol.

iv. The closed-world assumption on atomic formulas: the only true atomic formulas are those explicitly enumerated in (i).

For example, the following axioms constitute a vivid knowledge base:

i. P(a), P(b), Q(a), R(b,c).

ii. $a \neq b$, $a \neq c$, $b \neq c$.

iii. $\forall_X\ X = a \vee X = b \vee X = c$.

iv. $\forall_X\ P(X) \Rightarrow X = a \vee X = b$.
$\forall_X\ Q(X) \Rightarrow Q(a)$.
$\forall_{X,Y}\ R(X,Y) \Rightarrow X = a \wedge Y = b$.

Levesque makes the following important observations about vivid knowledge bases.

1. The actual implementation of the knowledge base need only record the ground sentences in (i), and these are easily recorded in a table. The sentences of types (ii), (iii), and (iv) can be left implicit in the inference machinery.

2. A vivid knowledge base is *complete*: It determines the truth or falsehood of any sentence in the first-order language.

3. The truth of any sentence in the language can be determined by directly checking whether it holds on the tables. No deeper kind of inference is required.

4. Let α be any first-order sentence with n quantifiers, and let m be the number of constants in the knowledge base. Then the truth or falsehood of α is computable in time $O(m^{n+1} \mid \alpha \mid)$.

5. A vivid knowledge base is *consistent.*

6. A vivid knowledge base is, in a sense, its own model. More precisely, a model of the theory can be constructed by picking the domain to be the set of all constant symbols, and picking the extension of each predicate to the set of all tuples of constants asserted in ground sentences (i) to satisfy the predicates. Moreover, this is the only model of the theory, up to isomorphism.

Clearly, a vivid knowledge base is very much like a database. However, the universal imposition of a closed-world assumption is harshly restrictive, even in database applications. Real database examples require a number of different forms of closed-world assumption. Some of these are explicitly expressible in a database language; others are left to the good understanding of the user. Consider, for example, a chronological academic database that records the record of each students since the system was installed in 1975; and consider the state of this database during the first week of registration, Fall 1991. Now, the predicate registered(STUDENT, CLASS, SEMESTER) satisfies the closed world assumption for SEMESTER = Spring, 1980; it does not satisfy the closed world assumption for SEMESTER = Fall, 1991; and it certainly does not satisfy the closed world assumption for SEMESTER = Spring, 1995. Thus, the question "Did Sarah Clark take Geology I in Spring '80?" can be answered true or false, while the question "Is Sarah Clark taking Geology I in Fall '91?" can be answered only true or unknown. Consider, next, the relation student(NAME, SEX, BIRTH, RACE), and let us suppose that the student has the option of specifying that his/her race should not be recorded in the database. The question, "Is Sarah Clark an American Indian?" can be answered either true (if she is so recorded), false (if she is recorded as being something else), or "Don't know" (if no race is recorded, or if Sarah has not yet been entered into the system.) The deduction of falsehood is justified by the fact that RACE depends functionally on NAME. The same remarks apply to the closed-world assumption on entities; the closed-world assumption would be probably be good for categories of race, but not for student names (if we include those that will arrive next year.) In short, even in very simple domains, property 2 above of completeness is not a feature but a bug.

For another example, consider a family tree containing the single predicate, "parent(X, Y)", containing both living and deceased persons. The strongest closure assumption that could be made about such a knowledge base would be that if X is mentioned in the knowledge base, and Y is a child of X who has been born, then "parent(X, Y)" is stated. Certainly it would not be reasonable to assume that all true parenthood relations are stated, or that the parents are recorded for every person mentioned, or that the unborn children of people are recorded.

In a knowledge base that uses either a modal or a second-order logic, it is not clear how the closed-world assumption should work. If the knowledge base records that John knows that A is on B and that John knows that B is on C, should we infer that John knows that nothing is on A, or that John does not know whether anything is on A?

The closed-world assumption on entities is also stronger than is usual in database theory. The relational calculus usually avoids making such an assumption by restricting the query language to "safe" queries, queries whose answer is not affected by the closed-world assumption on entities.

In general, database-style constraints, such as functional dependencies, multi-valued dependencies, cardinality constraints, and range constraints can neither be expressed nor enforced in a vivid representation. Thus, vivid representations are substantially weaker than databases.

A natural reaction would be to say, "All this goes to show is that you need a more powerful language. Suppose we add a database-like constraint language; where is the harm?" I think that this is, in fact, correct; but it is important to note that such an extension gives up most of the properties mentioned above. The representation of the knowledge base must now include the constraints, in addition to the atomic sentences, sacrificing properties (1), (3), and (6); the knowledge base is no longer complete, sacrificing property (2); the knowledge base is no longer "automatically" consistent, sacrificing property (5). Whether inference is still tractable (property (4)) depends on the language of constraints allowed.

Vivid representations may form the core of an AI representation system, but there must be more; there is a difference between AI and database theory. In his 1989 paper "Logic and the Complexity of Reasoning," Levesque proposes three further extensions to the theory. The first extension is to combine a vivid knowledge base with a simple inference capacity in the form of function-free Horn clauses with variables, that is, rules of the form

$$\mathrm{p}_1(X_1 \ldots X_k) \wedge \mathrm{p}_2(X_1 \ldots X_k) \wedge \ldots \wedge \mathrm{p}_n(X_1 \ldots X_k) \Rightarrow \mathrm{q}(X_1 \ldots X_k)$$

The closed-world assumption (iv) above is modified to specify that an atomic formula α is true only if it can be proven from the knowledge base. (Circumscribing over all of the predicates yields the same result.) The set of all true atomic formulas can be derived by starting with the vivid knowledge base and repeatedly adding the consequent of any Horn formula whose antecedent is satisfied until no new formulas can be derived.

Such a theory is called a *Horn* knowledge base. Properties (1), (2) (completeness), and (5) (consistency) still hold. Property (4) (efficiency) must be modified as follows: For any first-order sentence α, let m be the number of constants in the knowledge base, let n be the number of quantifiers in α, and let k be the maximum arity of any predicate. Then the truth of α can be determined in time $\mathrm{O}(m^{n+k+1} \mid \alpha \mid)$. Property (5) (self-modelling) holds in the weaker form that the set of provable atomic ground formulas is a model of the theory.

Levesque remarks in a footnote that the theory can handle function symbols as well, if the depth of self-nesting can be controlled. Here, however, there is an important caveat: in the above time estimate, m must be taken to be the number of *terms* constructible in the language. Thus, the time is no longer polynomial in the size of the language; it is exponential, and a pretty large exponential at that. Consider, for example, a planning program which deals with plans of up to ten steps, which has five actions types, each of which takes two arguments, and which works in a world of ten objects. There would then be $500^{10} \approx 10^{27}$ plans. The above formula would then give us a time bound of about 10^{10^8} steps to verifying a statement of the form "$\exists_P$ accomplishes(P,on(a,b))." Such a bound, of course, is wholly useless.

This is certainly not surprising; in fact, it's not even undesirable. Rather, a guarantee like property (4) above, that every expressible formula can be efficiently calculated, would also be a bug rather than a feature. Its contrapositive, after all, is that any fact that cannot be efficiently calculated cannot be expressed. Consider planning, again: We know[4] that, even in very limited theories of planning, finding a plan to satisfy a goal is an intractable problem that does not admit complete algorithms that are polynomial in the size of the plan or the size of the domain. A theory that guarantees that the truth of formulas can be verified in polynomial time must therefore be incapable of expressing the statement "Plan P achieve goal G."

[4] Assuming P$\neq$NP.

Levesque's second extension is to admit limited forms of disjunction. Intuitively, disjunctions are allowed if the disjunction contains essentially no more information than some more general atomic statement. For example, "John is seventy-one or seventy-three years old" can be allowed, if, relative to the theory, it carries no more information than "John is in his seventies" or "John is old." By contrast "John is seventy-one or Mary is twenty-three," is not allowed.

Formally, a knowledge base is in *semi-Horn* form if it can be divided into three parts:

1. A Horn knowledge base;

2. A set of disjunctions over atomic formulas not in (1);

3. For each disjunction $p \vee q$ in (2), a sentence of the form $(p \Rightarrow r) \wedge (q \Rightarrow r)$ where r is atomic.

No closed world assumption can be made, since there is no way of choosing between disjuncts. Semi-Horn knowledge bases, therefore, do not have the completeness property (2). They are consistent (property 5) and it is still possible to establish quickly, for any sentence, whether the sentence is true, false, or undetermined.

It may be noted that the example of turning disjunctions of quantities into intervals, such as turning "John is either 71 or 73 years old" to "John is between 71 and 73," is somewhat misleading. The translation does have the computational properties that Levesque suggests, but only in a theory that has pretty feeble ideas about intervals and ignores the fact that age is a function of the individual. For example, under the ordinary interpretation, the two facts "John is either 71 or 73" and "John is either 69 or 73" would have as a consequence "John is 73." It would not, however, be a consequence in any theory obeying Levesque's criteria. Another example: in an ordinary theory of intervals "John is between 70 and 74" is a consequence of "John is between 71 and 73"; inconsistent with "John is between 64 and 69"; and is consistent both with "John is between 68 and 72" and with its negation. This behavior, however, is not possible in Levesque's theory.

The final extension suggested by Levesque is to allow a limited degree of reasoning with explicit negation: namely, those inferences that are countenanced by relevance logic. The limitations of the resultant theory are now quite different from those discussed earlier; they are limitations of inferential power, rather than of expressive power. This suggestion, therefore, lies largely outside the scope of this paper. Without entering into a detailed discussion, let me say that this limited use of negation neither eliminates the issues discussed above, nor does it handle all the intuitively simple cases of reasoning with negation. For instance, the inference, "Either my office keys are in my pocket or I left them at home; they are not in my pocket; therefore I left them at home," is an intuitively simple inference that lies outside the scope of the theory. (Note that this is also a disjunction that cannot be subsumed in a larger category.)

Finally, Levesque suggests that human reasoning is of two kinds. The first is automatic, quick reasoning, which can be applied where the information is in tractable form; the second is "puzzle-mode" reasoning, which is deliberate and conscious and which must be applied when the information is presented in some confusing incomplete form. In this way he counters the argument of Moore [1982] that since people can solve problems involving any of the constructions of first-order logic, therefore the internal representation language must have at least the power of first-order logic.

To some extent, I agree with Levesque here. There is certainly an intuitive difference between automatically perceiving the consequences of an inference and painstakingly working it out. Also, it seems perfectly reasonable to suppose that a knowledge base ordinarily enforces substantial restrictions on expressivity in order to achieve efficiency. For example, it would be ill-advised to try to use a large cognitive map that maintained an arbitrary collection of sentences in a geometric language. Certainly, disjunctions between unrelated propositions seem like a promising candidate for exclusion.

However, Levesque and other advocates of lucid representations have gone too far in the other direction. An intelligent system may not have to use arbitrary forms of partial knowledge, but it must use some forms. In this paper, I have tried to avoid "puzzle-like" examples, and discuss only

inferences that seem to me "automatic". Of course, these are purely subjective judgements, but no one has suggested an operational criterion for distinguishing the two.

The best type of test for Levesque's theory would be to take a cognitive activity that is usually automatic, such as language understanding or visual interpretation, and consider the cases where these have to be consciously thought through, such as garden-path sentences. If these cases corresponded to violations of Levesque's criteria, that would be strong evidence for his theory. However, this does not seem to be the case.

In defense of Levesque, it should be said that he puts his ideas forward more as illustrations of a general direction for research, than as specific proposals to which he is committed. However, the discussion above indicates that there is little that can be accepted in Levesque's suggestions beyond the general observation that there should be some trade-off of expressivity for efficiency. We have found good reason to doubt that an AI theory of even the most natural and automatic forms of cognition can insist either on complete information, or on negation as failure, or on tractability of deriving arbitrary consequences of the theory.

3 Picture-like representations

In AI theories of spatial and physical reasoning, the demand for lucid representations is specialized to the demand that spatial representations be *picture-like.* The purpose of this section is to show that this goal is much more problematic than it initially appears. By "picture-like" I mean that the the representation could easily be used to construct a physical picture or model that expresses exactly the same spatial information, no more and no less. This definition has plenty of vagueness in it — what is a "physical picture" what does it mean to "use" the representation, how easy is "easily", what is the spatial information expressed by a physical picture — but it is clear enough that one can generally distinguish between representations that are more picture-like, and those that are less, which is all I will need.

Examples of picture-like representations would include occupancy arrays; polyhedra with numerical (floating-point) coordinates on the vertices; or constructive solid geometry, with precise shapes and numerical parameters. A spatial representations would not be not picture-like if it gives only a collection of topological constraints [Randell and Cohn, 89], [Kuipers, 77]; or if metric parameters are not given exactly, but only constrained to lie within intervals [Davis, 86] [McDermott and Davis, 84] or to obey symbolic constraints [Brooks, 81]; or if boundary curves or surfaces are only constrained only to lie within a certain region [Ballard, 81] [Requicha, 83], [Davis, 86].

Note that this definition is somewhat different from the "analogical" representations discussed by Sloman [1975]. For example, a specification of a collection of points by their coordinates would be picture-like, but not, by Sloman's definition, analogical. Conversely, an array of object names in decreasing order of weight would be a representation of weight that is at least weakly analogical in Sloman's definition; but it would not be picture-like, since there is no spatial information whatever involved.

3.1 Occupancy Array

In an occupancy array, each cell of the array corresponds to a single square in a rectangular tiling (or other type of tiling) of the plane, and each cell is labelled with some characteristic of the square. The label may be a visual property such as color of grey level; a name or a property of a single object occupying the square; a list of several regions that occupy the square; and so on. (Figure 1)

Occupancy arrays have many of the same favorable properties that are found in Levesque's vivid representations. It should be noted that the objections we brought to these properties in section 2 do not necessarily apply when we limit the discussion to spatial information.

1. Completeness. In a picture-like representation, any spatial property of objects, such as "A is less than 2 feet from B" or "C envelopes D," is determined. If we assume that all objects are

represented, then the representation determines the truth of any sentence in the language of geometry and objects, such as "There is a white object that is less than two feet from every green object that envelopes a red object."

2. Consistency. The geometric information is necessarily consistent. We do not have to worry that we have specified that some region is a round square, or that we have implicitly violated the triangle inequality. It is also easy to enforce certain physical rules, such as the rule that no two solid objects overlap.

3. Efficiency. Set operations, such as computing the union of two regions, or determining whether two regions overlap, are easy and fully parallelizable in an occupancy array.

The geometric information in an occupancy array is very nearly a vivid representation in Levesque's sense. It can be viewed as a collection of sentences of the form "occupies(X, Y, O)", object O occupies the square of coordinates $< X, Y >$. The array data structure, however, creates certain a number of important differences. It automatically enforces the sort and range restrictions on X and Y; X and Y cannot help being integers of the proper range. The rule that objects do not overlap is a functional dependency of O on X and Y, which, in an array, is either automatic or trivial to enforce, depending on the nature of the labels used. On the other hand, if the spatial field is largely empty, then the logical representation requires memory only proportional to the space actually occupied, while the array requires a constant amount of space.[5]

3.2 Limitations of the occupancy array

An occupancy array can represent directly only regions that are exactly equal to the union of squares aligned with the coordinate axes. There are two possible view of this limitation. The first is to view the representation as an exact description; all objects have a shape that is exactly the union of squares. The second is to view the representation as an approximation; the actual object does not really have all those corners, but it is very nearly the specified shape. These two views lead, in turn, to three different policies in writing procedures that operate on the representations:

1. Following the view that the representation is exact, the procedures might return the answers that would hold on those exact regions. In figure 2, for example, we would say that the distance from object A to B is exactly three units, and that the area of object A is exactly thirteen square units.

2. Following the view that an occupancy array is an approximation, the procedures may explicitly take cognizance of the approximation. In figure 2 we would say that the distance from object A to object B is between three and five units and that the area of object A is between five and thirteen square units.

3. The system refuses to address any question whose answer depends on details of size smaller than or equal to the grid size. In this way, it can avoid choosing between the two viewpoints. In figure 2, the question "What is the distance from A to B?" would have to be rephrased, "Give a value that is within 2 units of being the distance from A to B." It is not clear what it would be possible to ask about the area.

Adopting policy (2), of explicitly dealing with the approximations, means losing many of the desired properties of lucid representations. The representation is no longer complete. The computations may no longer be easy, either for the computer to perform, or for the programmer to work out. If the policy is to be followed consistently, the semantics of the representation must be carefully

[5] Unless quad-tree or sparse array representations are used. But then the enforcement of the consistency conditions becomes trickier.

defined; that is, it is necessary to establish a convention as to which representations can approximate which real shapes [Davis, 90]. For example, in figure 2, if we say that a square is marked as belonging to an object if the object at all overlaps the square, then the distance between A and B is between three and five units. If we say that a square is marked as belonging to an object if the object overlaps at least half the area of the square, then the distance between A and B is between two and four units.

Moreover, it may still emerge, after the semantics have been fixed, that there are shapes that cannot even be approximated. For example, a natural rule for viewing a region in an occupancy array as an approximation is to posit that the real region may partially fill boundary cells, but must completely fill internal cells. However, there is no way to describe a shape with deep thin holes under that convention.

If policy (2) is adopted, it becomes tempting to include an explicit expression of the degree of approximation in the representation. Adding a parametrized characterization of approximation will not usually add substantial further complications to the code, and will greatly increase its expressive power. Being able to vary precision is particularly welcome as imprecision tends to accumulate as inferences are performed.

Adopting policy (1), of constraining the microworld to regions that follow the representation exactly, means dealing with a very peculiar microworld. Consider, for example, the situation of four vertically stacked horizontal bricks shown in figure 3. In this situation, the two middle blocks can each independently slide horizontally without any of the other blocks moving. However, neither of the middle blocks can move in any other direction without moving one of the other blocks. The occupancy array representations displays these properties well. Now, rotate the structure by 45 degrees. Presumably, these kinematic properties should not be affected by the rotation. How can the rotation be expressed in an occupancy array so as to achieve this? If the rotated blocks are packed tight (figure 4A) then the middle ones cannot move in any direction. If they are packed with room to spare (figure 4B) then the middle blocks can move two units in the perpendicular direction, which is probably greater than tolerance. There are solutions that avoid both of these; for example, the blocks could be packed tight as in figure 4A and either move discontinuously, or move with a sort of slithering distortion in the boundary cells. However, defining the physics of such a world is not easy.

Policy (3), that one should never ask questions that depend on information more precise than the representation supports, looks at first glance like hard-nosed common sense. The shape is specified quite accurately, and nothing should depend on tiny imperfections. The problem is that many natural theories do depend on small details of shape. For example, two adjacent solid objects can be linked by hooks that can be almost arbitrarily small, if there are enough of them (Figure 5). Even basic geometric concepts become quite peculiar, if they must be converted to a form that is indifferent to small variations. (See, for example, the definition of the tolerance-space analogue of a straight line in [Kaufmann, 91].) Moreover, what kind of information the representation supports and what kind of queries are consequently acceptable depend on the same kind of delicate semantic issue discussed in connection with policy (2).

3.3 Difficulties in detailed spatial representations

CAD programs and similar applications use a rich spatial language, powerful specialized routines, and very detailed and precise spatial models. Not surprisingly, these representations avoid the crude problems encountered by occupancy arrays. However, the same underlying issues arise in subtler form, and they constitute major difficulties for the designers of solid modelling systems. In fact, these systems face the same choice of interpretive policies that we have discussed above, with similar trade-offs.

First, it should be noticed that fairly subtle geometric properties can make a physical difference, if a sufficiently broad class of physical phenomena is considered. For example, in manufacturing objects, it may be much easier to bore out a cylindrical hole (to within some given precision) than

a precise polyhedral approximation to a cylinder. Therefore a program that is working in this domain and that uses polyhedrons as a shape representation must adopt policy (2), and view its representations as approximations. In general, to design a program that will reason about a rich class of physical situations while sustaining the policy (1) that representations are exact, it will be necessary to use a very rich language of shapes. In modelling physical situations that give rise to shapes that are largely random or fractal, it will be almost impossible to sustain this policy.

Suppose, though, that we are truly committed, come hell or high water, to viewing spatial representations as exact. Thus, we want a representation that contains all the exact primitives we want; that can represent an arbitrary possible spatial layout to high (or arbitrary) precision; and whose range is closed under all the operations we want. Such representations do exist, for pretty extensive requirements. For instance, the class of all algebraic regions and algebraic transformations covers most shapes and operations of interest, and computational techniques exist to deal with it.

But such representations do not come cheap. In fact, they are not really what we originally meant by a lucid representation; they may be complete, but they are far from perspicuous. Basic questions such as "Is this shape description consistent?" "Is this shape connected / bounded / non-empty?" or "Do these two shapes overlap?" may be quite difficult to answer. Even the question of whether the representation is complete may be hard to answer. (The "wire-frame" representation of three-dimensional objects was in use for some time before it was observed that it is occasionally ambiguous.) These are certainly not representations whose meaning is directly obvious on looking at them.

Moreover, they are particularly subject to the problems of real arithmetic. (Hoffmann [1990] devotes the better part of a chapter to this issue.) Many geometric operations (e.g. finding distances or performing rotations by angles other than 90 degrees) gives rise to irrational numbers. These can be either treated symbolically or approximated. In practice, they are almost always approximated as floating point numbers, making round-off error a problem. Geometrical calculations are, for various reasons, particularly susceptible to the build-up of round-off error. After surprisingly few steps, error can accumulate to the point that calculations are meaningless, or even to the point that representations are inconsistent.

Thus, in the final analysis, we must either use a wholly symbolic and constraint-based system of reasoning, or we must view the representation as, ultimately, only an approximation to the exact shape and make the inference routines routines sensitive to the approximation. Either of these means that we are not dealing with a lucid representation. Both are similar to approaches that might be used for dealing with partial knowledge.

Of course, there are major qualitative differences between dealing with an initial uncertainty of one part in 10^{-16} (double precision CAD representation), one part in 10^{-3} (occupancy arrays), and one part in two (seriously partial information.) Many techniques that work well on the more precise representations die on the less precise. The key point, though, is that exact or near exact spatial representations sufficient for general physical reasoning are not easy; they are exceedingly hard. They are so hard that the incorporating partial information may not be tremendously more costly.

4 Simulations

The temporal analogue of a picture-like representation is a *simulation,* a complete representation of every state that holds and every event that occurs over a time interval. Prediction programs are often designed to output simulations; given an initial situation, or an initial situation and a completely specified plan of actions by an actor, they give a complete prediction of the future. Such programs generally construct their predictions by stepping forward through time, either using discrete rules that characterize the effect of a single action on a state, or using differential rules that characterize the change to a situation over differential time. Such inference techniques necessarily generate a complete time-line.

Another motivation for structuring the program so that the output is a simulation is to avoid the question of distinguishing between important and unimportant predictions. It is certainly easier for the programmer, and may be easier for the program, just to give everything, than to try to determine the appropriate focus for reasoning.

Qualitative reasoners [Weld and de Kleer, 89] generate predictions that are incomplete in significant respects. First, the characterization of states are partial; parameters are not given exact values but only constrained to lie within intervals. Second, the program does not construct a unique simulation, but rather a directed graph of states, and predicts that the true behavior must like on some path in this graph. Nonetheless, these program also work by applying inferences that take a single step forward through time, and therefore each path through the envisionment graph must contain every event and state change within the system that is expressible in the language of the system. (The system described in [Williams, 86] outputs a partial ordering of events and state changes, rather than a total ordering. However, all such events and state changes are enumerated.)

In the long run, however, prediction programs will have to be able to reason about significant events in the future without generating every minor event in between. McDermott [1989] gives the example of being asked to predict what would happen if someone leaves gasoline in the fireplace in July. A simulation approach would have to generate everything that happens in the house until someone decides to light a fire in December. The same problem can arise even in highly restricted systems. [Davis, 1988] studies the problem of deducing that a small die dropped inside a large funnel would eventually come out the bottom (Figure 6). To carry out a simulation would involve predicting every state of collision, spinning, sliding, and so on — very difficult to predict, ill-conditioned, unstable, high susceptible to numerical error, and useless.

5 Indirect Representations Despite Complete Information

Even where complete information is available initially, it may be helpful to use indirect representations in a reasoning program. The most important examples of this, as discussed in section 1, are where the inference process must go through intermediate steps, and where requiring a lucid representation for the intermediate level will cost more in the first stage of inference than it saves in the second.

Let me begin with a simple but artificial example, and then discuss more realistic cases. Suppose you are given the differential equation $\dot{y} = -e^{\sin(y)}$, $y(0) = 2$, and you wish to determine whether $y(t) = 5$ for any $t > 0$. An inference system based on lucid representations would find an exact solution for y and work from there. A more reasonable inference path would be to observe that $-e^{\sin(y)}$ is always negative; that y is therefore a monotonically decreasing function of time; and that therefore it is always less than 2 for $t > 0$. (It would easy to extend QSIM [Kuipers, 86] to make this inference.)

More realistically, the simulations discussed in section 4 mostly fall into this category. A full simulation of a complex event, like an automobile being driven off a cliff, can be enormously laborious. If only qualitative information is required, this labor is largely wasted. There are similar examples from spatial reasoning. If I am planning a route from my home in New York to an auditorium at Bar Ilan University, it may be helpful to know that the two locations are about seven thousand miles apart, but there is nothing whatever to be gained from knowing the distance to the inch.

There are also cases where including indirect representations in a knowledge base simplifies the process of inference. In particular, the inference path may be shorter from the indirect representation. In the trivial case, if ϕ is some indirect fact, it will be easier to derive ϕ from a knowledge base that contains it directly than from a lucid knowledge base. Similarly, it may well be easier to infer ϕ from a knowledge base containing ψ and $\psi \Rightarrow \phi$ than from a lucid knowledge base. For example, consider asking someone how many stars there are on the American flag. Even if he knows the layout of the flag, he will probably simply answer the question directly, or recall that the number of stars is equal to the number of states, which he knows to be fifty, rather than visualize the flag and count stars.

6 Partial Information and Lucid Representations

So far, we have largely restricted our discussion to cases where complete or nearly complete information is, in principle, available. Even in such cases, as we have shown, there are often cogent reasons to consider indirect representations. But the real case for indirect representations is that intelligent agents inescapably must often deal with partial information; and lucid representations, by definition, are ill-equipped to represent partial information.

Using complete information may be impossible or undesirable for a number of reasons: (Some of these have already been mentioned above.)

- Perception and text, the ultimate sources of information for an autonomous agent, generally give only partial information.[6]
- It may be impossible or very expensive to derive complete information of a desired type from inference rules.
- The agent may wish to reason about an underspecified system, such as a device that he is currently designing. If the designer can reason about the device in the middle of the design process, he may be able to catch and correct bugs at an early stage, and thus avoid much inefficiency.
- The agent may be able to save a great deal of work if he can find a general rule that applies to a class of similar situations, rather than working through the details of each one separately
- A complete representation may be too large. Requiring that a representation be complete means that all aspects of the world under consideration must always be described to the maximum degree possible. Consider, for example, a robot that can read going into a library. Since the robot can read, it must have a representation capable of expressing, directly or indirectly, that word XYZ is in position Q on page P of book B. In a complete representation, every such fact must be expressed; i.e., the robot must know the entire library. In a language with functional terms, or with self-embedding modal operators, such as "A knows ϕ", the number of atomic sentences may be unbounded.

The need to deal with partial information, however, does not absolutely rule out the possibility that most of the inference process can be carried out on lucid representations. It is conceivable that we could carry out reasoning from partial information by reducing it to reasoning on lucid representations, and thus avoid any need for indirect representations except in problem statements and answers. Such a reduction has been advocated by Levesque [1986] and others. In the remainder of this section we consider its feasibility.

Sometimes, it is possible to convert a collection of facts to a simpler representation in a sound, deductive way by weakening the information expressed. [Etherington et al., 89] discusses a number of ways in which this can be done. For instance, the disjunction "teacher(sam) $\vee$ professor(sam)" can be replaced by the vivid statement "instructor(sam)". (This is similar to Levesque's use of semi-Horn clauses, discussed in section 2, but it is performed in forming the knowledge base, rather than being explicitly represented in the knowledge base.) However, the resultant knowledge base is obviously still incomplete. Moreover, in many settings, particularly in spatial and physical reasoning, it will not be possible to carry out such a conversion without unacceptable losses in information.

Rather, it is generally accepted that using a lucid representation in reasoning from partial information will almost always involve *adding* information to the givens in order to make them lucid. Levesque [1986], for instance, discusses the following example. Given that Sam can hold only one object in each hand, and given that Sam first picks up a fork, then picks up a knife, infer that Sam cannot now pick up a spoon. Ordinarily, this inference would require reasoning by cases, considering

[6] Fleck [1988] stresses the need for reasoning programs to take account of the limited precision of measurements.

separately the case where Sam holds the fork in his right and the knife in his left, and the case where Sam holds the fork in his left and the knife in his right. Levesque, however, suggests that a reasoner can avoid the difficulties of reasoning by cases by using heuristic rules to vivify the description and choose between the disjuncts. In this case, one might apply a rule such as, "In the absence of other constraints, most people ordinarily use their right hand to pick up an object." Using this rule, the reasoner can make the assumption that Sam first picks up the fork with his right hand, and then, being constrained, picks up the knife with his left. The reasoner can then easily determine that Sam's hands are full.

Another example: Suppose we are given a description of a physical device that does not specify exact values for its numerical parameters, but only gives partial constraints. We wish to determine the behavior of the device. In order to do this, we pick some set of precise numerical values that satisfy the constraints. Now, since we have a complete description, it is much easier to calculate what the device will do. Having made this calculation, we can guess that the actual behavior of the device will be qualitatively similar to the calculated behavior, though exact numerical values will differ.

We can generalize these examples to the following general architecture (Figure 7): Starting with partial information, we *instantiate* the representation by adding enough information to pick out a specific lucid instance satisfying the partial constraints, We next draw conclusions using efficient inference techniques for lucid knowledge bases. Finally, we *abstract* from our conclusions those that are valid conclusions of the original information, discarding those that are artifacts of the information added during instantiation.

The middle step of this architecture — the drawing of inferences from a lucid representation — has been discussed above. Let us here grant its feasibility as an assumption. The two other steps, however, can be problematic. Section 6.1 will discuss the problems involved in instantiation. When is it reasonable to suppose that the process of instantiation is substantially easier than applying standard inference techniques to the given partial information? In Levesque's example, is it actually easier to apply the proposed default rule than to do the reasoning by cases? Section 6.2 will discuss the problems involved in abstraction. How can we distinguish between valid conclusions and artifacts of the instantiation? In Levesque's example, how can we distinguish between valid conclusions, such as, "Sam's hands are full," and invalid conclusions, such as, "Sam is holding the fork in his right hand," or "Sam can free his right hand by putting down the fork?"

In the case of an incremental knowledge base, which alternates assimilating new information with answering queries, such as an incremental knowledge base, we can imagine two ways in which lucid models can be used: (1) The knowledge base can be expressed as a set of indirect constraints. Lucid models could be used in the inference processes that answer queries from the knowledge base and that assimilate input information into the knowledge base. (2) The knowledge base could itself be a lucid model or a collection of lucid models. The second approach has a strong intuitive appeal and uses lucid models in a much more central position than the first. It relies critically, however, on having good heuristics for modifying a model in response to new information. Such heuristics are known for few domains.[7]

6.1 Instantiation

The first step of our proposed reasoning system is to replace the body of partial information by a lucid instance satisfying the partial constraints. It is important, therefore, that this instantiation be easy — certainly that it should be much easier than finding the desired conclusions directly from the given partial information.[8]

[7] Connectionist systems are close to (2). They rely, however, on having large numbers of inputs. Moreover, connectionist systems have primarily been studied in the context of learning from examples; it is not known whether they are suitable as an architecture for inference from partial knowledge.

[8] There are contexts where this need not be required. For instance, if it is desired to make a large number of inferences from a fixed body of constraints, it may make sense to invest a lot of time in finding an instantiation,

Certainly, this condition will sometimes hold. Our example above of instantiating the parameters of a physical device may be very well be such a case. We are substituting solving a static system of constraints, and performing a dynamic analysis using precise value for performing a dynamic analysis with constrained values. It is quite likely that we will come ahead on that trade-off.

However, there are many cases where instantiation is demonstrably not substantially easier than other types of inference. In many natural problems where inference is intractable, instantiation is also intractable. For example, given a set of Boolean formulas P, it is intractable to compute whether formula q is a consequence of P; but it is also intractable to find an instantiation of P, an assignment of truth values satisfying P. Similarly, given a constraint network N, it is just as hard[9] to find a set of values satisfying N as to determine whether an additional constraint c is a consequence of N.

Of course, how easy it is to find an instantiation depends on how broad a class of instantiations we allow. For example, the statement "Boolean formula q is a consequence of P" is equivalent to the statement, "$P \Rightarrow q$ is universally valid." Any truth valuation whatever can be considered an instance of the latter. The problem, of course, is that most are not very interesting instances; they make $P \Rightarrow q$ true just by making P false. As we shall see in section 6.2, what should be allowed as an instantiation depends on the goals and methods of the abstraction.

Levesque's suggestion that instantiation should be performed using default rules does not help, at least in our current (limited) understanding of default reasoning. As Kautz and Selman (1989) have shown, determining the consequences of a default theory tends to be difficult, even if the form of the default rules are limited.

If we try to make instantiation an initial step to *all* inference from partial information, we run into absurdities; instantiation may be much more difficult than some significant inferences. Two examples:

1. Given that the blocks in figure 8 are in configuration A at one time and in configuration B at a later time at a later time, infer that block X must have been moved in between. Here an instantiation would be a specific sequences of actions that change A into B.

2. Infer that the battery in a battery-powered automobile will require recharging from time to time. Here an instantiation would be a design for the automobile.

As these examples illustrate, instantiating partial information may require reasoning of practically any kind; planning in example (1), design in example (2). Requiring that these instantiations be complete before the given inferences be considered is obviously allowing a very small tail to wag a very large dog; inferences should be part of the planning/design process, not vice-versa.

6.2 Abstraction

Abstraction — the distinguishing between results that are genuine consequences of the given information and those that are artifacts of the particular instantiation — is the last and hardest step of the inference procedure. I know of two techniques for abstraction that apply across a wide range of domains and problems: proof guidance and Monte Carlo search. There are also a variety of techniques that apply only narrowly, to specific domains or problems; I will here examine symmetry arguments, proof of algebraic identities from examples, and numerical extrapolation.

6.2.1 Proof guidance

When a person is trying to construct a proof of a general result, he will often study the behavior of a particular instance, in order to "get a feeling" for the domain. Analogously, it is sometimes possible for an AI system to use results computed for a specific instantiation in order to guide the

or a number of instantiations, that will support rapid inference — more time than would be required for any single inference from the original constraints.

[9] Technically, the problem of instantiation in both these cases is NP-complete, while the problem of determining a consequence is co-NP-complete. On a deterministic machine, they take the same time in the worst case.

construction of a valid proof for some partial specifications. As a negative technique, this device was used as early as Gelernter's (1963) geometry theorem prover. Proposed lemmas of a desired theorem were checked against a specific diagram; if they did not hold on the diagram, there was no point in trying to prove them in general. Similarly, one can imagine an inference engine that suggests lemmas based on interesting properties computed on the instantiation. Consider, for example, a reasoning system that wants to infer that a block sliding on a horizontal surface with Coulomb friction will reach a state of rest in finite time. The reasoner might simulate the system for some specific values, and observe that the speed of the block decreased as a linear function of time till it stopped. It could then use the proposed lemma, "The speed decreases linearly," as an intermediate step to proving the result, "The block will stop."

This type of reasoning is similar to explanation-based learning. The key difference is that in EBL, one chooses the initial specifications to be the weakest that will support the conclusions; here, the initial specifications are given, and one chooses the strongest conclusions that they support.

This divide-and-conquer strategy seems potentially powerful and psychologically plausible as a model of proving theorems. Without very good search heuristics, however, it merely replaces a hard problem by a collection of harder ones; we must now evaluate a whole collection of proposed lemmas to determine, not only whether they are supported by the initial partial information, but also whether they are likely to serve as an intermediate step towards our goal.

One way to avoid this search problem is to require that the analysis of the instantiation be carried out in a form that can be directly generalized to a proof from partial specifications. This is analogous to explanation-based generalization in the narrow sense; the proof of the specific instance is variabilized to construct a general rule. The problem with this approach is a common one in EBG: the greater the simplification made by instantiating to a single example, the further the analysis of the instantiated example is from any analysis that can be carried out on the original specifications, and the less helpful the analysis is in guiding a general proof. If the analysis of the instantiation can be easily generalized, then the analysis of the partial specifications is probably not much more difficult than the analysis of the instantiation. In particular, this will often be true of numerical instantiation; it is not, in general, easy to generalize a result based on exact values to one based on partial constraints. (See, however, [Simmons, 88].)

6.2.2 Monte Carlo Search

Monte Carlo search is probably the most general technique for using lucid instantiations to reason with partial information. Rather than use a single instantiation, one generates a collection of instantiation; preferably, a collection that randomly samples the space of possible instantiations. A property that appears in only some of the instantiations is certainly not a genuine consequence of the given information; a property that appears in all the instantiations is a promising candidate.

Monte Carlo search is an effective procedure if the following conditions are met:

1. It is easy to generate a random instance satisfying the given constraints. This generally means either that the constraints have a relatively simple form, or that the solution space of the constraints occupies some substantial part of a larger simple space

2. It is much easier to compute the query on a large number of lucid instances than on a single collection of constraints.

3. Any important qualitative behavior occupies a substantial part of the search space.

4. If the answer is Boolean, then the difference between, "Almost certainly true," and "Certainly true," is not crucial.

For example, suppose one wishes to compute the range of values of a complicated continuous function of a number of numerical parameters $x_1 \ldots x_k$ subject to a collection of constraints of the form $a_i \leq x_i \leq b_i$ where a_i and b_i are given real numbers. Monte Carlo techniques typically work

very well in such settings. Condition (1) is always satisfied; one simply picks parameter values within the given ranges according to independent uniform distributions. Condition (2) is likewise almost always satisfied. Computing a complicated function of numerical values is relatively straightforward, while computing the exact bounds on such function on the cross-product of intervals can be very tricky. Condition (3) states that there are no small volumes of parameter space where the function takes on very different values from anywhere else. This is probably a safe assumption for most functions that arise in practice, though it is easy to exhibit functions that violate this, such as

$$e^{-(x_1^2+\ldots x_k^2)/\epsilon^2}$$

which is tiny everywhere outside a small neighborhood of the origin. Condition (4) does not apply, since this is not a Boolean query. (Monte Carlo techniques were used in this way for spatial reasoning in [McDermott and Davis, 84] and [Davis, 86].)

However, if these conditions are not met, then Monte Carlo search will not work. Condition (1) fails, obviously, if instantiation is difficult, as discussed in section 4.1. More importantly, even if it is easy to find an instantiation, it may be difficult to find a process that generates random instantiations. For example, as discussed above, the problem of instantiating a sequence of events that can turn a given starting state into a given ending state is the problem of plan construction. In some domains, constructing a single plan may not be too difficult; constructing a *random* successful plan (whatever that might mean), is a different question, however. It may not be sufficient to insert a randomizing element into the instantiation procedure.

Violations of condition (2) have been discussed in section 5.

Condition (3) often fails. Three examples:

1. Given that the blocks in figure 9 are in configuration A at one time and in configuration B at a later time, is it possible that block X0 moved in between? It seems plausible that most procedures to generate a "random" sequence of events interpolating between A and B will take a long time to find one that move X0. However, it is quite apparent that X0 could have moved, and a sensible reasoning procedure should be able to realize that. By contrast, if we specify that only a minute passes between A and B and that a motion takes 10 seconds, then it is apparent that X0 could not have moved. This illustrates condition (4); the distinction between "X0 probably did not move" and "X0 could not have moved" is apparent. (It is easy to construct examples where this distinction has practical importance; for example, suppose that X0 would have become contaminated in some horrible way if it had moved.)

2. A person is observed trying to open a cylinder lock of unknown characteristics with a key. Will it turn? Monte Carlo search in the space of cylinder lengths will take a long time to find the case where the key fits the lock. However, a reasoner should realize that in actuality the chances are good, since people generally use the correct key. It will not do to say that the reasoner will just use the rule "People generally use the correct key," and skip physical reasoning altogether. If the reasoner observes that the key is larger than the keyhole, then he ought to be able to deduce that the key will not turn.

3. In the theory of matrices, there are many simple procedures that work and theorems that hold on all non-singular matrices, or all matrices with no two multiple eigenvalues. The exceptions thus constitute a space of measure zero in the space of all matrices. However, in practice, matrices that are degenerate in one sense or another turn up much more frequently than this measure-theoretic argument would suggest, and these cases therefore cannot be ignored.

Even in cases where Monte Carlo search works well, there is something unappealing about it. Part of our motivation for using lucid models, after all, was that lucid models seem, intuitively, like what people use when they think about or visualize specific examples. A theory that postulates that we actually think using dozens or hundreds of specific instances seems far from one's introspective impression. Note that in Levesque's example of picking up a fork and knife, Monte Carlo search reduces to examining the two disjunctive alternatives, which is what we wanted to avoid in the first place.

6.2.3 Proof from a Single Example

Certain types of algebraic and geometric theorems can be proven or established to a high degree of certainty by showing that they hold on a single example. [P. Davis, 77], [Schwartz, 80].

Suppose that we want to establish an algebraic identity; for example, the equation

$$1 - x^8 = (1 - x)(1 + x)(1 + x^2)(1 + x^4)$$

One way to verify this is by multiplying it out symbolically. An alternative technique is to observe that this is an eighth-degree equation, which must either be an identity or have at most eight different roots. Therefore, if we check that it holds for nine different values, then it must hold in general. Moreover, given any eighth-degree equation that is not an identity, the probability is at most $8/N$ that a randomly chosen integer value between 1 and N is a root; the probability is zero that a randomly chosen real number between 0 and 1 is a root. Therefore, if we pick such a random number, or several such random numbers, and verify that they satisfy the equation, we can be reasonably sure that the equation is an identity.

The same technique can be applied to a multinomial equation like Euler's four-square identity

$$(a^2 + b^2 + c^2 + d^2)(w^2 + x^2 + y^2 + z^2) =$$

$$(aw - bx - cy - dz)^2 + (ax + bw + cz - dy)^2 + (ay - bz + cw + dx)^2 + (az + by - cx + dw)^2$$

If a multinomial equation like this is not an identity, the space of all its roots may have infinitely many points, but it defines a set of measure zero in the space of all possible values of the variables. Therefore we can test whether the equation is an identity by simply picking random values for the variables and checking whether the equation holds for these values. If it does not hold on the particular values, then we know that it was not an identity; if it does hold, then that forms very good evidence that the equation is an identity. Thus, we have a test that cannot give a false negatives and has probability zero of giving a false positive.

Many geometric theorems can be established in the same way. Consider the statement that the medians of a triangle meet in a point. This statement corresponds to some kind of algebraic identity on the coordinates of the vertices of the triangle. Therefore, it either holds for all triangles or for almost no triangles. We can therefore check it by picking three random points in the plane, and ascertaining that it works for these three points. If the statement were not universally true, the probability that we would happen to pick three points where it held would be zero. Note that we did not even have to figure out what the corresponding algebraic equation was; we just had to recognize that there was such an equation. The same technique, therefore, works for many standard geometric theorems: the coincidence of the altitudes of a triangle, the nine-point circle, Pappus' theorem, and so on.

In practice, of course, it is not feasible to pick random real numbers or to use them in computations. The mathematical literature on this technique has therefore studied methods that are computationally feasible and that give low probabilities of false probabilities. For example, it is possible to show that if one chooses random integer values between 1 and N, then the probability of chancing on a set of roots of the equation decreases at least as fast as $1/N$ [Schwartz, 80]. Moreover, the results of different random choices are independent; hence, if k different sets of values and prime bases are chosen, the probability of a false positive decreases at least as fast as N^{-k}. Also, the k different computations can be performed in parallel.

This techniques, however, cannot be applied to inequalities; an inequality such as $X < Y$ holds over half of any region symmetric in X and Y. Likewise, it does not apply to the problem of inferring one algebraic equation from another; the equation $X = Y$ holds on half the roots of $X^2 = Y^2$. Unfortunately, inequalities are inescapable in most commonsense applications of quantitative inference, particularly in physical reasoning and in cognitive mapping. The qualitative characteristics of a physical system or of a spatial environment are largely determined by inequalities rather than by exact equations. It seems unlikely, therefore, that these techniques will be broadly applicable in AI systems.

6.2.4 Symmetry

If the problem statement exhibits some kind of symmetry, then valid conclusions must have the same kind of symmetry. This rule can be used in Levesque's example. The proposed conclusion "Jack must be holding the fork in his right hand," can be rejected by observing that the original problem specifications are symmetric between the two hands, and that therefore any valid conclusion must likewise be symmetric.

Somewhat more subtly, if the process of instantiation involves only a choice among symmetric alternatives, then any symmetric inference from that instantiation must be generally valid. Thus, since instantiation in Levesque's example involves only the choice between "John picks up the fork with his right hand" versus "with his left hand," we may conclude that the symmetric conclusion "Both John's hands are occupied," is, in fact, a valid conclusion independent of the instantiation.

6.2.5 Numerical Extrapolation

In numerical problems, it is sometimes possible to predict the behavior of a function in an interval from its behavior at a single point. For example, if $f(X)$ is continuous and differentiable, then the values it attains in the interval $[x_0-\epsilon, x_0+\epsilon]$ cannot lie far outside the interval $[f(x_0)-\epsilon\dot{f}(x_0), f(x_0)+\epsilon\dot{f}(x_0)]$. One can use such estimation techniques to approximate the range in which a constraint on $f(X) < c$ can be expected to hold. If a universal constraint on the magnitude of the derivative can be established, then the conclusion can be made with certainty [Hong and Tan, 90].

6.2.6 Summary of Abstraction Techniques

These techniques for abstracting general conclusions from examples are interesting and useful. However, it is clear that they do not in general suffice for the desired range of commonsense inference from partial information, particularly when combined with the difficulties in instantiation discussed in section 6.1. It is certainly possible that more powerful techniques will be developed to enable this style of inference, but surely the burden of proof is on those who claim that this can be done.

7 Conclusions

The key observations made in this paper are the following:

- Even in simple database-like applications, it is not generally appropriate to require that a knowledge base be complete. Rather, a rich language for expressing completeness and integrity conditions should be available. In wider contexts, the requirement that inference be efficient (polynomial-time) is similarly inappropriate.
- All known spatial representations that suffice for a broad range of physical reasoning either encounter the issues of incomplete knowledge and imperfect approximations or involve very complex symbolic reasoning.
- Effective reasoning about an extended course events will require abandoning the use of simulations, in the sense of complete time lines.
- Even in cases where complete information is in principle available, it may be wise not to restrict the representation to be lucid.
- The use of partial information cannot be avoided in intelligent autonomous systems. The proposal that reasoning on partial information can be implemented via lucid representations is extremely problematic; the techniques known for doing this are of very limited application.

It must be admitted that in the current state of the art, lucid representation are often more practical than indirect representations. This is particularly true as regard spatial and temporal representations; in existing programs, much more power can be gotten from picture-like representations and simulations than from any known partial representations. However, there is a large and ever-growing body of knowledge about classes of partial knowledge where complete inference is tractable, and about partial but useful types of partial inferences from broader classes of partial knowledge. There is, then, no need to despair of the possibility of combining efficiency with a large degree of expressivity. The cheerful assumption that lucid representations will always suffice for the needs of intelligent systems, and that indirect representations can always be avoided, seem to me without basis; and a narrow focus on those tasks that can be achieved using lucid representation will lead — indeed, has already led — to unrealistic limits on the types of inference that are considered.

8 References

Ballard, D., "Strip Trees, A Hierarchical Representation for Curves," *Communications of the ACM,* vol. 24, no. 5, 1981, pp. 310-321.

Brachman,R., H. Levesque, and R. Reiter (eds.) *Proc. First Intl. Conf. on Principles of Knowledge Representation and Reasoning,* Morgan Kaufmann, San Mateo, CA, 1989.

Brooks, R., "Symbolic Reasoning among 3-D models and 2-D images,;; *Artificial Intelligence,* vol. 17, nos. 1-3, 1981, pp. 285-348.

Davis, E., *Representing and Acquiring Geographic Knowledge,* Pitman Press, London, 1986.

Davis, E, "A Framework for Qualitative Reasoning about Solid Objects," in [Weld and de Kleer, 89], pp. 603-610.

Davis, E., *Representations of Commonsense Knowledge,* Morgan Kaufmann, San Mateo, CA, 1990.

Davis, P.J., "Proofs, Completeness, Transcendentals, and Sampling," *Journal of the ACM,* 1977, vol. 24, pp. 298-310.

Dean, T. and M. Boddy, "Reasoning about Partially Ordered Events," *Artificial Intelligence,* vol. 36, 1988, pages 375-387.

Doyle, J. and R. Patil, "Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services," *Artificial Intelligence,* vol. 48, 1991, pages 261-297.

de Kleer, J., "Multiple Representations of Knowledge in a Mechanics Problem Solver," *Proc. IJCAI-77,* pp. 299-304.

Etherington, D.W., A. Borgida, R.J. Brachman, H. Kautz, "Vivid Knowledge and Tractable Reasoning: Preliminary Report," *ICJAI-89*, 1146-1152.

Faltings, B., "Qualitative Kinematics in Mechanisms," *IJCAI-87,* pp. 436-442,

Fleck, M., "Boundaries and Topological Algorithms," MIT Tech. Report AI-TR-1065, 1988.

Forbus, K., P. Nielsen, and B. Faltings, "Qualitative Kinematics: A Framework," *ICJAI-87*, pp. 430-435.

Funt, B., "Problem-Solving with Diagrammatic Representations," in R. Brachman and H. Levesque (eds.), *Readings in Knowledge Representation,* Morgan Kaufmann, 1985, pp. 441-456.

Gardin, F. and B. Metzler, "Analogical Representations of Naive Physics," *Artificial Intelligence,* vol. 38, no. 2, 1989, pp. 139-160.

Gelernter, H., "Realization of a Geometry-Theorem Proving Machine," in E. Feigenbaum and J. Feldman, (eds.) *Computers and Thought*, McGraw-Hill, 1963.

Gelsey, A., "Automated Reasoning about Machine Geometry and Kinematics," *Proc. IEEE Conf. on AI Applications,* 1987, pp. 182-187.

Halpern, J. and M. Vardi, "Model Checking vs. Theorem Proving: A Manifesto," in J. Allen, R. Fikes, and E. Sandewall (eds.) *Proc. Second Intl. Conf. on Principles of Knowledge Representation and Reasoning,* 1989, pages 325-334.

Hoffman, C., *Geometric and Solid Modelling,* Morgan Kaufmann, San Mateo, CA, 1990.

Hong, J. and X. Tan, "Proving Inequalities by Example," Comp. Sci. Dept. New York University, Unpublished, 1990.

Joscowicz, L., "Shape and Function in Mechanical Devices," *Proc. AAAI-87,* pp. 611-616.

Kaufmann, S., "A Formal Theory of Spatial Reasoning," in J. Allen, R. Fikes, and E. Sandewall (eds.) *Proc. Second Intl. Conf. on Principles of Knowledge Representation and Reasoning,* 1989, pages 347-356.

Kautz, H. and B. Selman, "Hard Problems for Simple Default Logics," in [Brachman, Levesque, and Reiter, 89], pages 189-197.

Kosslyn, S., *Image and Mind,* Harvard University Press, 1980.

Kuipers, B., "Modelling Spatial Knowledge," *Cognitive Science,* vol. 2 no. 2, 1978.

Kuipers, B., "Qualitative Simulation," *Artificial Intelligence,* vol. 29, 1986, pp. 289-338.

Levesque, H., "Making Believers out of Computers," *Artificial Intelligence*, vol. 30, pp. 81-108, 1986.

McDermott, D., "Artificial Intelligence Meets Natural Stupidity," in J. Haugland (ed.) *Mind Design,* MIT Press, 1981.

McDermott, D. "Mental Models," unpublished abstract, 1989.

McDermott, D.V. and E. Davis, "Planning Routes through Uncertain Territory," *Artificial Intelligence,* 1984.

Miller, D., "Planning by Search through Simulations," Yale University Research Report #423, 1985.

Moore, R., "The Role of Logic in Knowledge Representation and Commonsense Reasoning," *Proc. AAAI-82.*

Pylyshyn, Z., *Computation and Cognition: Toward a Foundation for Cognitive Science,* MIT Press, 1984.

Randell, D. and A. Cohn, "Modelling Topological and Metrical Properties in Physical Processes," in [Brachman, Levesque, and Reiter, 89].

Requicha, A.A.G., "Toward a Theory of Geometric Tolerancing," *The International Journal of Robotics Research,* vol. 2, no. 4, 1983, pp. 45-60.

Schwartz, J.T., "Probabilistic Algorithms for Verification of Polynomial Identities," *Journal of the ACM,* 1980.

Simmons, R., "A Theory of Debugging Plans and Interpretations," AAAI-88, pp. 94-99.

Slade, S., "Case-based Reasoning: A Research Paradigm," *AI Magazine,* vol. 12, no. 1, 1991, pages 42-55.

Sloman, A., "Afterthoughts on Analogical Representations," *Proceedings of Theoretical Issues in Natural Language Processing,* Cambridge, MA 1975. Reprinted in Brachman and Levesque (eds.) *Readings in Knowledge Representation,* Morgan Kaufman, 1985.

Soderland, S. and D. Weld, "Evaluating Nonlinear Planning," Tech. Rep. #91-02-03, Dept. of Computer Science, University of Washington, 1991.

Tversky, B., "Distortion in Memory for Maps," *Cognitive Psychology,* vol. 13 no. 3, 1981, pp. 407-433.

Weld, D. and J. de Kleer (eds.), *Qualitative Reasoning about Physical Systems,"* Morgan Kaufmann, San Mateo, CA, 1989.

Williams, B., " Doing Time: Putting Qualitative Reasoning on Firmer Ground," *AAAI-86,* pp. 105-113.

Woods, W., "What's in a Link: Foundations for Semantic Networks," in D. Bobrow and A. Collins (eds.) *Representation and Understanding,* Academic Press, 1975.