The Tractability of Subsumption in Frame-Based Description Languages

Ronald J. Brachman
Hector J. Levesque

Fairchild Laboratory for Artificial Intelligence Research
4001 Miranda Avenue
Palo Alto, California 94304

ABSTRACT

A knowledge representation system provides an important service to the rest of a knowledge-based system: it computes automatically a set of inferences over the beliefs encoded within it. Given that the knowledge-based system relies on these inferences in the midst of its operation (*i.e.*, its diagnosis, planning, or whatever), their computational tractability is an important concern. Here we present evidence as to how the cost of computing one kind of inference is directly related to the expressiveness of the representation language. As it turns out, this cost is perilously sensitive to small changes in the representation language. Even a seemingly simple frame-based description language can pose intractable computational obstacles.

1. Introduction

There are many different styles of knowledge representation system in use in Artificial Intelligence programs, but they all have at least this in common: the representation system is supposed to provide both a repository for the beliefs of the knowledge-based system in which it is embedded, as well as automatic inferences over those beliefs. Typical inferences automatically computed by AI representation systems include inheritance of properties, set membership and set inclusion, part/subpart inferences, type subsumption, and resolution.

Here we address a fundamental problem in the nature of the service to be provided by knowledge representation systems: the greater the expressiveness of the language for representing knowledge, the harder it becomes to compute the needed inferences (see [7] for an overview of this tradeoff). In this brief paper, we present a formal analysis of the computational cost of expressiveness in a simple frame-based description language. We illustrate how great care needs to be taken in the design of a representational facility, even when our intuitions about the language tell us that it is a simple one. As it turns out, even an apparently modest representation language can prove intractable.

2. Subsumption in Frame Languages

Among the more popular representation languages in use today are those based on the notion of *frames* (see, for example, [1], [3], and [9]). Frames give us the ability to define structured types; typically a frame comprises a set of more general frames (its *superframes*) as well as a set of descriptions of the attributes (*slots*) of instances of the frame. The most common type of slot description specifies a restriction on the value of the filler of the slot for all instances of the frame. The restriction can be as specific as a particular value that all instances of the frame must exhibit (alternatively, the value may be just a *default*, in which case an individual inherits the value provided he does not override it), or it may be a more general constraint on attribute values, in which case this *value restriction* is usually a pointer to another frame. Less commonly, the number of required fillers is also specified in a slot restriction (often in terms of a minimum and a maximum number of attribute values).[1] The generalization relation between frame and superframe, or between two frames where one is simply a more restricted version of another, implicitly forms a *taxonomy*, or *inheritance hierarchy*.

Notationally, a frame might be defined by a list of superframes (with either an explicit or implicit "*isa*" relation [2]), followed by a set of slot restrictions expressed by attribute/value-description pairs (with attribute and value-description usually separated by a colon). For example, the simple frame

```
[PERSON
     child (≥ 1):
     son:            LAWYER
     daughter:       DOCTOR]
```

is intended to be a structured type representing the concept of a person that has at least one child, and all of whose sons (*i.e.*, male children) are lawyers and all of whose daughters are doctors. Similarly, the more complicated frame,

```
[STUDENT, FEMALE
   department:  COMPUTER-SCIENCE
   enrolled-course (≥ 3) :
             [GRADUATE-COURSE
                  department:  ENGINEERING-DEPARTMENT]]
```

is intended to be a structured type that describes female Computer Science students taking at least three graduate courses in a department within a school of Engineering.[2]

There is a natural correspondence between this frame form of description and noun phrases in natural language. For example, the above frame might just as well have been written as "**student** and a **female** whose **department** is **computer-science**, and who has at least 3 **enrolled-courses**, each of which is a **graduate-course** whose **department** is an **engineering-department**." A simple set of translation rules would allow us to move easily from frame form to (almost) readable English.[3]

[1]While the use of number restrictions is not widespread, they have been used extensively in KL-ONE [3] and languages like it [4]. They seem to be a useful generalization of the existential reading of slots (see below), so we include them here.

[2]Typically, frames are given *names* as well; for example, we might have labeled our first example frame, "PROUD-PARENT". We have explicitly chosen to avoid these here so as to eliminate confusion about their meanings. For this paper, we are interested in relations among frames implied by their *structure* only (see below), and will assume that atoms are all independent.

[3]For example, the list of superframes would translate into a conjunction of nouns ("**student** and a **female**"). A slot that had only one filler might translate into a simple "whose" clause ("whose **department** is **computer science**"). And a slot

One interesting property of these structured types is that we do not have to state explicitly when one of them is below another in the taxonomy. The descriptions themselves implicitly define a taxonomy of *subsumption*, where type A subsumes type B if, by virtue of the form of A and B, every instance of B must be an instance of A. In other words, it can be determined that being an A is implicit in being a B, based only on the structure of the two terms (no "user" needs to make an explicit statement of this relationship). For example, without any world knowledge, we can determine that the type "person" subsumes the type

"person each of whose male friends is a doctor",

which in turn subsumes

"person each of whose friends is a doctor whose specialty is surgery."

Similarly, "person who has at least 2 children" subsumes "person who has at least 3 male children".

The computation of analytic relations like subsumption (and others, such as *disjointness*—see [4]) is arguably the most important service to be provided by a frame description system (see [4] for evidence of this). If this service is to be provided in a reasonable fashion to the rest of a knowledge-based system, then these relations must be determined in a timely way. Thus, while expressive power is typically the most immediate concern of representation language designers, it cannot be addressed without simultaneous consideration of its computational implications.

Computational cost concomitant with expressive power has been treated in depth in the arena of formal languages like that of first-order logic. However, while frames have been used extensively in AI systems, and have been found expressively adequate for some tasks, their instrinsic computational properties have not been accounted for. We have explored the complexity of determining subsumption in a family of frame-based description languages, and have found that it is in fact remarkably sensitive to what seem to be small changes in the representational vocabulary. In order to illustrate this surprisingly touchy tradeoff, we here examine in detail a representative frame language and a simple variant.

3. A Formal Frame Description Language

Let us consider a simple description language, $\mathcal{FL}$, with two major syntactic types—*concepts* and *roles*. These will correspond to the typically less well-defined notions of "frame" and "slot". Intuitively, we think of concepts as representing individuals, and roles as representing relations between individuals. $\mathcal{FL}$ has the following grammar:

$$\begin{aligned}\langle concept\rangle ::= &\ \langle atom\rangle\\ &\mid (\textbf{AND}\ \langle concept_1\rangle \ldots \langle concept_n\rangle)\\ &\mid (\textbf{ALL}\ \langle role\rangle\ \langle concept\rangle)\\ &\mid (\textbf{SOME}\ \langle role\rangle)\end{aligned}$$

$$\begin{aligned}\langle role\rangle ::= &\ \langle atom\rangle\\ &\mid (\textbf{RESTR}\ \langle role\rangle\ \langle concept\rangle)\end{aligned}$$

While the linear syntax is a bit unorthodox, $\mathcal{FL}$ is actually a distillation of the operators in typical frame languages. Atoms are the names of primitive (undefined) concepts. **AND** constructions represent conjoined concepts, so for example, (AND adult male person) would represent the concept of something that was at the same time an adult, a male, and a person (*i.e.*, a man). In general, x is an (**AND** c_1 c_2 … c_n) iff x is a c_1 and a c_2 and … and a c_n. This allows us to put several properties (*i.e.*, superconcepts or slot restrictions) together in the definition of a concept. The **ALL** construct provides a value- or type-restriction on the fillers of a role (x is an (**ALL** r c) iff each r of x is a c). Thus (ALL child doctor) corresponds to the concept of something all of whose children are doctors. It is a way to *restrict* the value of a slot at a frame. The **SOME** operator guarantees that there will be a least one filler of the role named (x is a (**SOME** r) iff x has at least one r). For instance, (AND person (SOME child)) would represent the concept of a parent. This is a way to *introduce* a slot at a frame. Note that in the more common frame languages, the **ALL** and **SOME** are not broken out as separate operators, but instead, either every slot restriction is considered to have *both* universal and existential import, or exclusively one or the other (or it may even be left unspecified).[4] Our language allows for arbitrary numbers of role fillers, and allows the **SOME** and **ALL** restrictions to be specified independently. Finally, the **RESTR** construct accounts for roles constrained by the types of their fillers, *e.g.*, (RESTR child male) for a child who is a male, that is, a son (in general, y is a (**RESTR** r c) of x iff y is an r of x and y is a c).

It is simple to map more standard notations into our frame language. One reading of the the frame used as the first example in this paper is "person with at least one child, and each of whose sons is a lawyer and each of whose daughters is a doctor". In $\mathcal{FL}$, that reading would be represented this way:

```
(AND person
     (SOME child)
     (ALL (RESTR child male) lawyer)
     (ALL (RESTR child female) doctor))
```

4. Formal Semantics

We now briefly define a straightforward extensional semantics for $\mathcal{FL}$, the intent of which is to provide a precise definition of subsumption. This will be done as follows: imagine that associated with each description is the set of individuals (individuals for concepts, pairs of individuals for roles) it describes. Call that set the *extension* of the description. Notice that by virtue of the structure of descriptions, their extensions are not independent (for example, the extension of (**AND** c_1 c_2) should be the intersection of those of c_1 and c_2). In general, the structures of two descriptions can imply that the extension of one is always a superset of the extension of the other. In that case, we will say that the first *subsumes* the second (so, in the case just mentioned, c_1 would be said to subsume (**AND** c_1 c_2)).

Let $\mathcal{D}$ be any set and $\mathcal{E}$ be any function from concepts to subsets of $\mathcal{D}$ and roles to subsets of the Cartesian product, $\mathcal{D} \times \mathcal{D}$. So

$$\mathcal{E}[c] \subseteq \mathcal{D} \quad \text{for any concept } c, \text{ and}$$
$$\mathcal{E}[r] \subseteq \mathcal{D} \times \mathcal{D} \quad \text{for any role } r.$$

We will say that $\mathcal{E}$ is an *extension function* over $\mathcal{D}$ if and only if

1. $\mathcal{E}[(\textbf{AND}\ c_1\ \ldots\ c_n)] \ = \ \bigcap_i \mathcal{E}[c_i]$
2. $\mathcal{E}[(\textbf{ALL}\ r\ c)] = \left\{x \in \mathcal{D} \mid \text{if} \quad \langle x,y\rangle \in \mathcal{E}[r] \quad \text{then} \quad y \in \mathcal{E}[c]\right\}$
3. $\mathcal{E}[(\textbf{SOME}\ r)] = \left\{x \in \mathcal{D} \mid \exists y \left[\langle x,y\rangle \in \mathcal{E}[r]\right]\right\}$
4. $\mathcal{E}[(\textbf{RESTR}\ r\ c)] = \left\{\langle x,y\rangle \in \mathcal{D} \times \mathcal{D} \mid \langle x,y\rangle \in \mathcal{E}[r] \text{ and } y \in \mathcal{E}[c]\right\}.$

Finally, for any two concepts c_1 and c_2, we can say that c_1 *is subsumed by* c_2 if and only if for any set $\mathcal{D}$ and any extension function $\mathcal{E}$

with multiple fillers might translate into a "who (or that) has n" construct ("who has at least 3 enrolled-courses"), possibly followed by an "each of which" qualification ("each of which is a graduate-course"). Finally, a slot with multiple fillers, but with no number restriction specified, would translate simply into an "each (or all) of whose" qualification ("all of whose daughters are doctors").

[4] See [6] for some further discussion of the import of languages like KRL. As it turns out, the unversal/existential distinction is most often moot, because most frame languages allow only single-valued slots. Thus the slot's meaning is reduced to a simple predication on a single-valued function (*e.g.*, the slot/value pair age:integer means *integer(age(x))*).

over $\mathcal{D}$, $\mathcal{E}[c_1] \subseteq \mathcal{E}[c_2]$. That is, one concept is subsumed by a second concept when all instances of the first—in all extensions—are also instances of the second. From a semantic point of view, subsumption dictates a kind of necessary set inclusion.

For an illustration of how this is an appropriate view of subsumption, let us consider two descriptions in $\mathcal{FL}$, d_1 and d_2, where d_1 subsumes d_2:

```
d1 = (AND person
          (ALL child doctor))

d2 = (AND
       (AND person
            (ALL child rich))
       (AND male
            (ALL (RESTR child rich)
                 (AND doctor
                      (SOME (RESTR specialty surgery))))))
```

d_1 corresponds to "**person** each of whose **children** is a **doctor**," and d_2 corresponds to "**person** each of whose **children** is rich, and a **male** each of whose **rich children** is a doctor who has a **surgery specialty**."

A proof that d_1 subsumes d_2, based on our formal definition of subsumption, might go as follows. Let $\mathcal{D}$ be any set, $\mathcal{E}$ any extension function over $\mathcal{D}$, and x any element of $\mathcal{E}[d_2]$. By applying (1) above to d_2 twice, we know that $x \in \mathcal{E}$[**person**] and that by (2), if $\langle x,y\rangle \in \mathcal{E}$[**child**], then $y \in \mathcal{E}$[**rich**], and so by (4), $\langle x,y\rangle \in \mathcal{E}$[(**RESTR child rich**)]. Also, by (2), if $\langle x,y\rangle \in \mathcal{E}$[(**RESTR child rich**)], then, by (1) and the definition of d_2, $y \in \mathcal{E}$[**doctor**]. Putting these two together, we have that if $\langle x,y\rangle \in \mathcal{E}$[**child**], then $y \in \mathcal{E}$[**doctor**]. Since $x \in \mathcal{E}$[**person**], then by (2) and (1), $x \in \mathcal{E}[d_1]$. To summarize, because all of the children of a d_2 are rich, and each of a d_2's rich children is a certain kind of doctor, then all of d_2's children are doctors. Because any d_2 is also a person, the description d_2 is subsumed by the description d_1.

5. Determining Subsumption

Given a precise definition of subsumption, we can now consider algorithms for calculating subsumption between descriptions. Intuitively, this seems to present no real problems. To determine if a subsumes b, what we have to do is make sure that each component of a is "implied" by some component (or components) of b, exactly the way we just determined that d_1 subsumed d_2. Moreover, the type of "implication" we need should be fairly simple since $\mathcal{FL}$ has neither a negation nor a disjunction operator.

Unfortunately, such intuitions can be nastily out of line. In particular, let us consider a slight variant of $\mathcal{FL}$—call it $\mathcal{FL}^-$. $\mathcal{FL}^-$ includes all of $\mathcal{FL}$ except for the **RESTR** operator. On the surface, the difference between $\mathcal{FL}^-$ and $\mathcal{FL}$ seems expressively minor. But it turns out that it is computationally very significant. In particular, we have found an $O(n^2)$ algorithm for determining subsumption in $\mathcal{FL}^-$, but have proven that the same problem for $\mathcal{FL}$ is intractable. In the rest of this section, we sketch the form of our algorithm for $\mathcal{FL}^-$ and the proof that subsumption for $\mathcal{FL}$ is as hard as testing for propositional tautologies, and therefore most likely unsolvable in polynomial time.

Subsumption Algorithm for $\mathcal{FL}^-$: SUBS?[a,b]

1. Flatten both a and b by removing all nested **AND** operators. So, for example,

   (**AND** x (**AND** y z) w) becomes (**AND** x y z w).

2. Collect all arguments to an **ALL** for a given role. For example,

   (**AND** (**ALL** r (**AND** a b c)) (**ALL** r (**AND** . X))) becomes

   (**AND** (**ALL** r (**AND** a b c . X))).

3. Assuming a is now (**AND** $a_1 \ldots a_n$) and b is (**AND** $b_1 \ldots b_m$), then return **true** iff for each a_i,
   (a) if a_i is an atom or a **SOME**, then one of the b_j is a_i.
   (b) if a_i is (**ALL** r x), then one of the b_j is (**ALL** r y), where SUBS?[x,y].

The property of SUBS? that we are interested in is the following:

Theorem 1: SUBS? *calculates subsumption for $\mathcal{FL}^-$ in $O(n^2)$ time.*

Before considering a proof of the correctness of this algorithm, notice that it operates in $O(n^2)$ time (where n is the length of the longest argument, say). Step 1 can be done in linear time. Step 2 might require a traversal of the expression for each of its elements, and step 3 might require a traversal of b for each element of a, but both of these can be done in $O(n^2)$ time.

Now, on to the proof that this algorithm indeed calculates subsumption: first we must show that if SUBS?[a,b] is true then a indeed subsumes b (soundness); then we must show the converse (completeness). Before beginning, note that the first two steps of the algorithm do not change the extensions of a and b for any extension function, and so do not affect the correctness of the algorithm.

To see why the algorithm is sound, suppose that SUBS?[a,b] is **true** and consider one of the conjuncts of a—call it a_i. Either a_i is among the b_j or it is of the form (**ALL** r x). In the latter case, there is a (**ALL** r y) among the b_j, where SUBS?[x,y]. Then, by induction, any extension of y must be a subset of x's and so any extension of b_j must be a subset of a_i's. So no matter what a_i is, the extension of b (which is the conjunction of all the b_j's) must be a subset of a_i. Since this is true for every a_i, the extension of b must also be a subset of the extension of a. So, whenever SUBS?[a,b] is **true**, a subsumes b.

The proof of the completeness of the algorithm is a bit trickier. Here we have to be able to show that anytime SUBS?[a,b] is **false**, there is an extension function that does not assign a to a superset of what it assigns b (*i.e.*, in some possible situation, a b is not an a). There are three cases to consider, and for each of them we will define an extension function $\mathcal{E}$ over the set $\{0, 1\}$ that has the property that 1 is in the extension of every description, but 0 is in the extension of b but not that of a.

1. Assume that some atomic a_i does not appear among the b_j. Let $\mathcal{E}$ assign the ordered pairs $\{\langle 0,1\rangle, \langle 1,1\rangle\}$ to every role and $\{0,1\}$ to every atom except a_i to which it assigns $\{1\}$.
2. Assume that a_i is (**SOME** r), which does not appear among the b_j. Let $\mathcal{E}$ assign $\{0,1\}$ to every atom and $\{\langle 0,1\rangle, \langle 1,1\rangle\}$ to every role except r, to which it assigns only $\{\langle 1,1\rangle\}$.
3. Assume that a_i is (**ALL** r x), where if (**ALL** r y) appears among the b_j, then, by induction, x does not subsume y. Let $\mathcal{E}^*$ be an extension function not using 0 or 1 but such that some object $*$ is in the extension of y but not of x. Then, let $\mathcal{E}$ contain $\mathcal{E}^*$ and assign $\{0,1\}$ to every atom and $\{\langle 0,1\rangle, \langle 1,1\rangle\}$ to every role except r, to which it assigns $\{\langle 1,1\rangle, \langle 0,*\rangle\}$.

In all three cases it can be shown that $\mathcal{E}[a]$ is not a superset of $\mathcal{E}[b]$, and so, that a does not subsume b when SUBS?[a,b] is **false**. So, in the end, SUBS? is correct, and calculates subsumption in $O(n^2)$ time.

We now turn our attention to the subsumption problem for full $\mathcal{FL}$. The proof that subsumption of descriptions in $\mathcal{FL}$ is intractable is based on a correspondence between this problem and the problem of deciding whether a sentence of propositional logic is implied by another. Specifically, we define a mapping π from propositional sentences in conjunctive normal form to descriptions in $\mathcal{FL}$ that has the property that for any two sentences α and β, α logically implies β iff $\pi[\alpha]$ is subsumed by $\pi[\beta]$.

Suppose $p_1, p_2, \ldots, p_m$ are propositional letters distinct from A, B, R, and S.

```
π[p₁ ∨ p₂ ∨ ... ∨ pₙ ∨ ¬pₙ₊₁ ∨ ¬pₙ₊₂ ∨ ... ∨ ¬pₘ] =
    (AND (ALL (RESTR R p₁) A)
            ...
         (ALL (RESTR R pₙ) A)
         (SOME (RESTR R pₙ₊₁))
            ...
         (SOME (RESTR R pₘ)))
```

Assume that $\alpha_1, \alpha_2, \ldots, \alpha_k$ are disjunctions of literals not using A, B, R, and S.

```
π[α₁ ∧ α₂ ∧ ... ∧ αₖ] =
    (AND (ALL (RESTR S (SOME (RESTR R A))) B)
         (ALL (RESTR S π[α₁]) B)
            ...
         (ALL (RESTR S π[αₖ]) B))
```

A proof that this mapping has the desired property is given in [8]. What this means is that an algorithm for subsumption can be used to answer questions of implication by first mapping the two sentences into descriptions in $\mathcal{FL}$ and then seeing if one is subsumed by the other. Moreover, because π can be calculated efficiently, any *good* algorithm for subsumption becomes a good one for implication.

The key observation here, however, is that there can be no good algorithm for implication. To see this, note that a sentence implies $(p \wedge \neg p)$ just in case it is not satisfiable. But determining the satisfiablity of a sentence in this form is NP-complete [5]. Therefore, a special case of the implication problem (where the second argument is $(p \wedge \neg p)$) is the complement of an NP-complete one and so is a co-NP-complete problem. The correspondence between implication and subsumption, then, leads to the following:

Theorem 2: *Subsumption for $\mathcal{FL}$ is co-NP-complete.*

In other words, since a good algorithm for subsumption would lead to a good one for implication, subsumption over descriptions in $\mathcal{FL}$ is intractable.[5]

6. Conclusion

The lesson here is clear—there seems to be a sudden and unexpected "computational cliff" encountered when even a slight change of a certain sort is made to a representation language.[6] We are actively engaged in examining other dimensions of representation languages, in an effort to understand exactly what aspect of the representation is responsible for the computational precipice.

Besides warning us to be careful in selecting operators for a knowledge representation language, the tradeoff between expressiveness and computational tractability serves as admonition against blind trust of our intuitions. The change from $\mathcal{FL}^-$ to $\mathcal{FL}$ seemed simple enough, yet caused subsumption to become intractable. Other generalizations to $\mathcal{FL}^-$ that we have considered appear at least as dangerous, and yet in the end prove no problem at all. For example, we have examined a variant of $\mathcal{FL}^-$ that generalizes the **SOME** operator to be an **AT-LEAST** operator, whereby we can require any number of fillers for a certain role. Further, we might add an operator called **ROLE-CHAIN**, that allows us to string roles together. Given these two new operators, we could form interesting concepts like "a person with at least two grandchildren":

```
(AND person
     (AT-LEAST (ROLE-CHAIN child child) 2))
```

Remarkably enough, even the simultaneous addition of both of these operators to $\mathcal{FL}^-$ does not cause subsumption to fall off of the computational cliff [8].

This line of research probably has a long way to go before the definitive story is told on the complexity of computing with AI description languages. However, we seem to have made a significant start in formally analyzing an essential frame language and its variants. Further, the methodology itself is an important factor. Crucially, the notion of subsumption in this account is driven from the semantics, so that there is always a measure of correctness for the algorithms we design to compute it. Thus, we will not fall prey to one problem that has plagued work in this area since its inception—the excuse that what subsumption (or any other important relation) means is "what the code does to compute it". In fact, our approach so well defines the problem that we can find cases where it is provable that *no* algorithm of a certain sort can be designed.

Acknowledgements

This research was done in the context of the KRYPTON project, and as a result, benefited greatly from discussions with Richard Fikes, Peter Patel-Schneider, and Victoria Pigman.

REFERENCES

[1] Bobrow, D. G., and T. Winograd, "An Overview of KRL, a Knowledge Representation Language." *Cognitive Science*, Vol. 1, No. 1, January, 1977, pp. 3–46.

[2] Brachman, R. J., "What **IS-A** Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks." *IEEE Computer, Special Issue on Knowledge Representation*, October, 1983, pp. 30–36.

[3] Brachman, R. J., and J. G. Schmolze, "An Overview of the KL-ONE Knowledge Representation System." *Cognitive Science*, forthcoming.

[4] Brachman, R. J., R. E. Fikes, and H. J. Levesque, "Krypton: A Functional Approach to Knowledge Representation." *IEEE Computer, Special Issue on Knowledge Representation*, October, 1983, pp. 67–73.

[5] Cook, S. A., "The Complexity of Theorem-Proving Procedures." *Proc. 3rd Ann. ACM Symposium on Theory of Computing*. New York: Association for Computing Machinery, 1971, pp. 151–158.

[6] Hayes, P. J., "The Logic of Frames." In *Frame Conceptions and Text Understanding*. D. Metzing (*ed.*), Berlin: Walter de Gruyter & Co., 1979, pp. 46–61.

[7] Levesque, H. J., "A Fundamental Tradeoff in Knowledge Representation and Reasoning." *Proc. CSCSI-84*, London, Ontario, May, 1984, pp. 141–152.

[8] Levesque, H. J., and R. J. Brachman, "Some Results on the Complexity of Subsumption in Frame-Based Description Languages." In preparation.

[9] Minsky, M., "A Framework for Representing Knowledge." In *Mind Design*, J. Haugeland (*ed.*). Cambridge, MA: MIT Press, 1981, pp. 95–128.

[5]More precisely, the co-NP-complete problems are strongly believed to be unsolvable in polynomial time.

[6]It should be emphasized that the question of tractability is a matter of expressiveness, and not of the particular description language. Here we have used a simple language to illustrate our point, but the tradeoff affects any language that allows the same distinctions to be made.