Using Model Theory to Specify AI Programs

Alan M. Frisch

February 1985
Revised May 1985

Abstract

This paper proposes a method for adapting the traditional devices of model theory to the task of specifying the input/output behavior of artificial intelligence reasoning programs when viewed as inference engines. The method is illustrated by specifying two programs, one a toy example and the other a program for retrieving information from a declarative knowledge base. Close examination shows that many intuitions about the properties of a retriever can be stated rigorously in terms of inference and that the model-theoretic specification can then be used to prove that the retriever has these properties. The paper concludes by discussing some of the benefits that could be obtained by the use of model-theoretic specifications.

A condensed version of this paper will appear in Proceedings of the Ninth International Joint Conference on Artificial Intelligence, August, 1985.

Cognitive Science Research Paper

Serial no: CSRP 042

The University of Sussex
Cognitive Studies Programme
School of Social Sciences
Brighton, BN1 9QN
England

The success of artificial intelligence as a science hinges on our ability to build a theory that relates a program's structure to its behavior. Essential to this enterprise are methods for producing rigorous specifications of programs at a high level of abstraction that can be used to codify and communicate our results.

There are two specification methods predominantly (though certainly not exclusively) used: English prose typically describing various structural components of the program and their role in the program's performance, and the actual code that is used to implement the program. (For the sake of argument, call the latter LISP code.) An English prose specification can be highly abstract but, in practice, usually at the expense of rigor and precision. Consider the problem of predicting how a program so described in a journal will behave on examples not considered in the article. Worse yet, consider the problem of showing that the program has certain properties. Though a LISP-code specification does not suffer from the lack of precision that an English specification typically does, it is not sufficiently abstract. Hence, appropriately enough, LISP code rarely works its way into the literature. When is the last time you published a program? A less serious and more-easily remedied problem is that a LISP-code specification may need to be accompanied by a specification of LISP itself. I know of a researcher who misunderstood a published LISP program because he and the author had different versions of LISP in mind.

A methodology that has been pursued successfully throughout computer science is that of separating the description of <u>what</u> a program computes from <u>how</u> it computes it. On the one hand there are descriptions of a program's input/output behavior and on the other descriptions of its internal modules, processes, states and data structures. Descriptions of how a program works can and should be given at various levels of abstraction. For example, at one level a program can be described as a non-deterministic algorithm moving through a search space from an initial state to a goal state, while a less abstract description would specify a search strategy for realizing the non-deterministic algorithm on a deterministic machine.

This paper considers the case for using model theory to specify what a program computes. A method is proposed for adapting traditional model-theoretic techniques and is illustrated by specifying two programs. The first is a toy example used to illuminate the key points of the technique, while the second, a knowledge retriever, is used to demonstrate application of the technique to a realistic AI program. Beyond what is demonstrated by these examples and those mentioned in the section on related work, little is known about the range of applicability of the technique.

2. Overview of the Proposed Technique

A major concern of AI is with programs that manipulate representations, which I take to be data structures that denote. This raises the possibility that many such programs can be viewed as inference engines, deriving conclusions from their representations. This paper is concerned with developing techniques for using model theory to specify the input/output behavior of programs seen from this viewpoint.

For example, a planner can be seen as an inference system. The program embodies a theory of action and its input is a pair of sentences each predicating a condition on the world. The planner produces a plan such that its theory of action logically implies that the second input sentence would be true if the plan were performed in any world satisfying the first input sentence.[1]

Unfortunately, many planning systems such as STRIPS (Fikes and Nilsson, 1971) do not meet such a specification. In order to deal with the enormously complex problem of finding an appropriate plan among the set of all plans, these systems employ problem representations that often find plans quickly but do so at the expense of occasionally failing to find any at all. As an inference engine such a system is incomplete. How then can such an incomplete system be specified?

---

[1] The idea that plans are counterfactuals was brought to my attention by Andy Haas.

The approach advocated here is to produce another model theory whose logical implication relation is weakened[2] in such a way that the program is a sound and complete inference engine with respect to it. Many people initially find this approach quite odd. They are accustomed to thinking of a model theory as specifying what can be concluded validly from what--in some sense, as a competence theory of inference. I suggest that those who are comfortable with this viewpoint consider the weaker model theory as a performance theory of inference. Other people are accustomed to thinking of a model theory as a way of assigning meaning to symbols and are skeptical of producing a new meaning assignment. But I am not suggesting that the original model theory be discarded; on the contrary, it is still a valuable device in the study of meaning. The new model theory can be thought to provide an additional meaning assignment. If the program is working under this alternative meaning then it is a complete inference engine. Hence, the symbols mean one thing to us and something different to the program. According to our theory of meaning the program is incomplete but according to its weaker theory of meaning it is complete.

How can these new, weaker model theories be produced and what is their relationship to the unweakened model theory? To answer the question consider a model theory as laying down a set of constraints on what constitutes a model. Of all (mathematical) objects, only those that satisfy the constraints qualify as models. A model theory also associates with each model a truth assignment, a total function from sentences to their truth values. Hence, a model theory constrains the range of truth assignments that can be generated. In a standard propositional logic, for example, these constraints ensure that any truth assignment that assigns two sentences true, also assigns their conjunction true. The logical implication relation associated with a model theory is a product solely of the range of truth assignments that the model theory generates. Relaxing the constraints produces a new model theory, one that may generate additional truth assignments. No matter how the

---

[2] One logical implication relation is weaker than another if the inferences sanctioned by the first are a subset of those sanctioned by the second.

constraints are relaxed, the new model theory must have a weaker logical implication relation than the original. That is, if $|=_1$ and $|=_2$ are logical implication relations and $|=_2$ is obtained by relaxing the model theory for $|=_1$, then alpha $|=_2$ beta implies alpha $|=_1$ beta. To see this, observe that a truth assignment can serve only as a counterexample to a claim that one sentence logically implies another; hence if none of the truth assignments from the relaxed model theory are counterexamples then certainly none from the original model theory are.

3. A Toy Example

Consider a program that reasons about an arbitrary equivalence relation named "r". A user communicates with the program, making assertions and queries, each of which specifies a sentence of the form "r(alpha,beta)", where alpha and beta are symbols drawn from some lexicon.

This program can be specified in terms of the symbolic manipulations it performs. There are many conceivable specifications but for the sake of argument let us say that the program works by maintaining a collection of disjoint sets. Initially there is a unit set for each symbol in the lexicon. Whenever the user asserts "r(alpha,beta)" the program combines the sets that contain alpha and beta into a single set. To respond to the query "r(alpha,beta)" the program simply determines whether the elements alpha and beta are in the same set. There are many well-studied algorithms for performing this set-union task (Tarjan and van Leeuwen, 1984), the best of which can process a series of n assertions and queries in slightly greater than 0(n) time.

To the user of the system this specification is too detailed and too concrete. He does not need to know, nor does he care, whether the program works one way or another. At the level of abstraction with which the user is concerned the various implementations are all identical. Of course, for other concerns, such as implementation, there is a world of difference between different specifications. A more abstract, non-procedural definition of this reasoning system can be obtained by replacing the question "What does the reasoner do?" with "Given a set of sentences that have been asserted, what query sentences succeed?" At this higher level of abstraction the various set-union algorithms are

all equivalent.

In response to the question of what queries succeed, it can be shown that the program described above answers "yes" to a query if, and only if, it legitimately can do so based on what it has been told. That is, it answers "yes" if, and only if, the queried sentence logically follows from the set of asserted sentences. Forgive my pedantry while I spell out the obvious details of the model theory that gives rise to the logical implication relation for this language; these details will be valuable in considering how to relax a model-theoretic specification.

Each of the model theories discussed in this paper is given a name. The one presented next is called "E". In cases where confusion could arise, terms like "E-model" and "E-logical implication" and symbols like "$|\overset{=}{}_{r}$." are used to indicate which model theory is under consideration.

An E-model is a pair <D,A> where D is a non-empty set of individuals called the domain and A is a function that maps every symbol in the lexicon to an element of D and maps r to a binary relation over D such that:

(1) A(r) is reflexive,
(2) A(r) is symmetric, and
(3) A(r) is transitive.

The truth assignment associated with <D,A> is the function that takes each sentence of the form r(alpha,beta) to True if the relation A(r) holds between A(alpha) and A(beta), and to False otherwise. These truth assignments can then be used in the usual fashion to define the notions of E-satisflability, E-validity, and E-logical implication for this language.

This model theory serves two purposes in analyzing the program. First, it provides a rigorous semantics for the language that the program manipulates and in doing so defines logical implication for the language. Second, it is used in specifying what the program computes by stipulating that it responds "yes" if, and only if, the queried sentence E-logically follows from the asserted sentences. In the case of this program the turn pegs cm hanH-in-hand hpnancp the nrncrram IQ a cnnnri anrl

these two uses of a model theory as we turn our attention to a reasoning program that is not complete.

Suppose that for some reason we were not happy with a program that required slightly greater than O(n) time to process a series of n assertions and queries. (I told you this was a toy example!) Furthermore, suppose that we were willing to replace the set-union algorithm with the following algorithm, which is much weaker but slightly faster. Whenever "r(alpha,beta)" is asserted, the program adds the pair <alpha,beta> to an associative store. The program responds "yes" to the query "r(alpha,beta)" if, and only if, alpha and beta are identical or the associative store contains either <alpha,beta> or <beta,alpha>.

This program is incomplete with respect to E. For example, if only r(a,b) and r(b,c) have been asserted, the query r(a,c) will not result in "yes" even though the queried sentence E-logically follows from the two asserted sentences. E still gives a semantics for the language manipulated by the program but it no longer specifies what the program computes. However, there is a weaker model theory--call it E^W--whose logical implication relation does specify the input/output relation of this program. E^W is identical to E except that constraint (3), which says that A(r) must be transitive, is eliminated. With respect to E^W the program is a sound and complete inference engine--though admittedly soundness and completeness are normally taken to be with respect to a model theory that specifies the meaning of the language.

Every model in E is also a model in E^W, but not vice-versa. For example, consider the model $\langle D_1, A_1 \rangle$ where $D_1 = \{1,2,3\}$ and A_1 satisfies:

$A_1(a) = 1$
$A_1(b) = 2$
$A_1(c) = 3$
$A_1(r) = \{\langle 1,1\rangle, \langle 2,2\rangle, \langle 3,3\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle\}$

This model respects constraints (1) and (2) but not constraint (3). The truth assignment generated by $\langle D_1, A_1 \rangle$ is not generated by any E-model, hence $|=_{E^W}$ is strictly weaker than $|=_E$. Returning to the example previously cited, $\langle D_1, A_1 \rangle$ demonstrates that r(a,b) and r(b,c) do not E^W-logically imply r(a,c).

Though this example program specification is quite simple it has illustrated many of the major points about the method of specifying programs with model theory. The next section uses this method in the study of knowledge retrieval. Since a method should be judged by the results that can be obtained with it, the next section is not as concerned with the method itself as with how it can contribute to the study of retrieval.

4. Knowledge Retrieval

Artificial intelligence reasoning systems commonly employ a knowledge base module (KB) that stores information expressed in a representation language and provides facilities for other modules of the system to retrieve this information. Though there has been a growing concern for formalization in the study of knowledge representation, little has been done to formalize the retrieval process. This section outlines an attempt to use the proposed specification technique to remedy the situation. Successively, the four subsections that comprise this section

- show how retrieval can be viewed as a certain form of inference,[3]
- show how this form of inference can be given a model-theoretic specification,
- present some of the properties of the specified retriever,
- and discuss a couple of extensions that could be made to the specification.

4.1. Retrieval as Inference

The retrieval problem that I have studied takes the knowledge base to be a set of sentences of the first-order predicate calculus (FOPC). While FOPC may be less expressive than many other languages that could be used, it is expressive enough to lead to a serious retrieval problem; most notably, logical implication is only semi-decidable.

---

[3] Frisch and Allen (1982) present the case in more detail.

A query asks whether a specified closed sentence of FOPC can be retrieved from the KB, and the retriever responds "yes" or "no." So, for example, one could query "Can 'UNCLE(JOHN,BILL)' be retrieved?" It is not difficult to extend this notion of query to include FOPC sentences with free variables. Such an extension enables one to ask, "What are all the x's such that 'UNCLE(x,BILL)' can be retrieved?" However, for purposes of this exposition it suffices to consider only the first form of query.

A specification must determine whether a retriever responds "yes" or "no" for any given KB and any given query. Just as one can speak of a sentence logically following, or being provable, from a set of sentences, one can speak of a queried sentence being retrievable from a set of sentences contained in a knowledge base. Thus the task of specifying a retriever comes down to one of specifying a retrievability relation.

I place three requirements on any retrievability relation that I study. If kb is a set of sentences and q a single sentence of some language whose logical implication relation is $\models$ then any retrievability relation for that language should satisfy:

soundness: if q is retrievable from kb then kb $\models$ q.

verbatim retrieval: if q is in kb then q is retrievable from kb.

decidability: retrievability is a decidable relation.

The first requirement demands that sentences not logically following from the KB are not retrievable while the second demands that sentences explicitly in the KB are retrievable. The third requirement, which ensures that the retriever can be realized by an effective procedure that is guaranteed to terminate, is weaker than it ideally should be--that the retriever could be realized by a procedure requiring only some small amount of computational resources.

It should come as no surprise that I specify a retrievability relation for FOPC by identifying it with the logical implication relation of a model theory obtained by relaxing the standard Tarskian model theory for FOPC (Tarski, 1935). I call the Tarskian model theory "T" and the relaxed model theory that specifies the retriever "R".

Consider how, as a retrievability relation, $\models_R$ stands up to the above

requirements. Since $|=_R$ is obtained by relaxing constraints in the specification of $|=_T$, the soundness requirement is met, according to the argument of Section 2. Any logical implication relation will satisfy the second requirement, regardless of the mode] theory. Since $|=_T$ is not decidable, it is the third requirement that forces a relaxation of T and leads to the viewpoint of knowledge retrieval as limited (i.e., incomplete) inference. Hence it is the undecidability of FOPC that makes the retrieval problem interesting.

4.2. Specifying a Retriever

I have used this viewpoint of retrieval as limited inference to specify a slightly simplified version of the knowledge retriever incorporated in the ARGOT dialogue-participation system (Allen, Frisch and Litman, 1982). The specification is produced in stages, first by specifying a retriever that is extremely conservative in what it infers and then by extending it with additional forms of decidable inference such as those dealing with taxonomic hierarchies and property inheritance (as typically done by semantic-network systems). At each stage a retrievability relation is specified by identifying it with both a provability and a logical implication relation. A point worth noting is that these specifications of ARGOT's retriever were developed after the program. Only the extremely conservative retriever is considered here, though Section *4.4* briefly outlines how semantic-network-style taxonomic inference can be incorporated.

The strategy for ruling-out certain inferences to obtain this extremely weak inference engine is based on the important intuition that retrieval is more like a matching operation than a deductive operation. Stated in terms of inference, the simple retriever satisfies the no-chaining restriction; imagining the KB and query divided into quanta called "facts," the simple retriever cannot chain two facts together in order to respond affirmatively to a queried fact. In other words, a queried sentence is retrievable only if each of its facts is retrievable from a single fact in the KB. This intuition contrasts with the notion that retrieval deduces the queried sentence by repeatedly combining facts together to derive new facts.

This no-chaining restriction reduces the problem of deciding retriev bility for arbitrary sentences to that of deciding retrievability facts. This reduced problem can be kept simple by defining facts such a way that they themselves are simple.

The use of the word "fact" has been deliberately vague and it will n be made precise until Section 4.3. However before turning to t specification of retrieval it is worth noting that the main connecti in a fact is disjunction. This is a crucial point because R is deriv by weakening only the interpretation of disjunction. Hence, a fact given a weaker interpretation in R, eliminating chaining and thus maki R-logical implication decidable over the set of facts. A key examp worth bearing in mind is that the specification does not sanction a si ple form of chaining, the disjunctive form of modus ponens:

$$P, \sim P \vee Q \not\models_R Q$$

It is tempting to try to produce the specification of the relaxed mod theory by following the tactic used in Section 3 of simple textual del tion of some constraint on what constitutes an unrelaxed model. Ho ever, in the case of T it is not so straightforward. E specifies thr constraints on the relations that can be assigned to the symbol "r" a thus prevents the atomic sentences of the language from obtaining ce tain combinations of truth values. (Recall that all sentences in t object language of E are atomic.) Relaxing E to obtain E^W involv deleting one of the three constraints, allowing the atomic sentences be given additional combinations of truth values.

Unlike E, T places no constraints on the relations that a model c assign to a relation symbol and therefore the atomic formulas of t language can be assigned any combination of truth values. So the str tegy of generating additional truth assignments by giving the atom sentences more combinations of truth values cannot be pursued in th case.[4] Since the truth assignments to atomic formulas cannot

---

[4] It is assumed here that only 2 truth values are used. Beln (1975, 1977) pursues a strategy of using 4 truth values.

increased we must consider the trutn assignments to moiecuiar rormuias. T Is truth-functional; the truth value of a molecular formula in a model is a function of the truth values of the formula's constituents in that model. For example, a disjunction is true jn a T-model if, and only if, one of its disjuncts is true in that model. In order to admit more truth assignments while still maintaining a compositional semantics, the truth value of a molecular formula must be a function of some other feature of the formula's constituents. The best-known non-truth-functional model theories are the possible-worlds model theories (Kripke, 1963), which are commonly used for modal logics. In a PWMT (possible-worlds model theory) one speaks of the truth value of a formula in a world, and for a molecular formula this may be a function of the truth values of the formula's constituents in other worlds. There are many reasons for moving from a Tarskian to a possible-worlds model theory, but the sole motivation here is that the possible-worlds framework is a more expressive medium for describing a model-theory; every truth-functional model theory can be written in a possible-worlds form but not vice-versa.

So, the first step to reaching the ultimate target of a model theory for specifying the retriever is to produce a PWMT—call it T'—whose models correspond to those in T and which therefore yields precisely the Tarskian truth assignments. The next step is to produce R by relaxing T^f so that it allows more models and associates non-truth-functional truth assignments with the added models.

For pressent purposes, consider a model in a PWMT to be a 3-tuple <D,A,W> where D—the domain—and W—the set of worlds in the model—are non-empty sets and A is a function from a non-logical symbol and a world to an appropriate denotation for that symbol. Because the non-logical symbols may have different denotations in different worlds, there is a truth assignment associated with each world in a model. The manner in which such a truth assignment is derived varies from one PWMT to another.

The current task is to define T', a PWMT that corresponds to T. Since a T-model generates only one truth assignment, let T^1 be a possible-worlds theory in which all models contain exactly one world. Hence, the T'-model <D,A,{w}> corresponds to the T-model <D,lambda x.A(x,w)>. In the

obvious way, w in <D,A,{w}> can be associated with a truth assign
the same as that associated with its corresponding T-model. I make
explicit because, as previously mentioned, it is the manner in
these assignments are generated that is to be relaxed. Specific
consider the equation that inductively defines the way T^f assigns
values to disjunctions. If $V_{fit,w}$ is the truth assignment associated
world w in model m then

(4) $V_{m,w}(\text{alpha v beta}) = OR(V_{mw}(\text{alpha}), V_{m\,w}(\text{beta}))$,

where OR is the function such that

OR(True,True) = True
OR(True,False) = True
OR(False,True) = True
OR(False,False) = False

To specify R, T' is relaxed to allow models with more than one v
This enables R to construct non-truth-functional truth assignmer
defining the assignment associated with one world in terms of
associated with other worlds. In particular, R assigns True to *a*
junction in a world if any of its disjuncts are assigned True ir
world in the model. This modification is effected by replacing (4)
(5).

(5) $V_{m,w}$(alpha v beta) = True if for some world u in *m*
$OR(V_{m,u}(\text{alpha}), V_{m,u}(\text{beta})) =$
= False otherwise

The equation gives a formula of the form "alpha v beta" the
interpretation as a traditional PWMT would give to the modal fo
"possibly (alpha v beta)."

Notice that (4) and (5) are in full agreement on those models that
tain only a single world. Hence R still generates all the Tax
truth assignments. However, R also generates many non-truth-funct
truth assignments. As an example consider any two-world model
worlds are complementary—they disagree on the truth value assigns
every atomic formula. Every literal (an atomic formula or its negs

must be true in exactly one of the two worlds, and therefore in either world every disjunction of literals is true, regardless of the truth of the literals in that world. This vividly illustrates how the logic is weakened; if one knows only that a disjunction of literals is true, one knows nothing of the truth values of those literals.

R-models with complementary worlds play an important role in studying the properties of this model theory. Whereas R-models with identical worlds produce only the Tarskian truth assignments, those with complementary worlds are the most un-Tarskian in that their truth assignments differ most from the Tarskian truth assignments. Such R-models can demonstrate, for example, that P and ~P v Q do not R-logically imply Q. A counterexample is produced by a world that satisfies P and falsifies Q and is in an R-model that contains its complementary world.

4.3. Properties of the Retriever

This section examines the properties of the retrievability relation specified by model theory R. I state these properties without proof, concentrating instead on how they coincide with certain informal intuitions about retrieval.

The principal motivation for relaxing T to produce R was to obtain decidability of logical implication. Yet because R still gives negation, conjunction and quantification their standard Tarskian interpretations, $\models_R$ remains undecidable. However, it is decidable for sentences of a particular normal form. A <u>universal clause</u> is a universally quantified disjunction of literals. An <u>existential disjunction</u> is an existentially quantified conjunction, each conjunct of which is a disjunction of literals; that is it is of the form,

$$(E\ x_1\ x_2\ \ldots)\ \ (L_{11}\ v\ L_{12}\ v\ \ldots)\ \&\ (L_{21}\ v\ L_{22}\ v\ \ldots)\ \&\ \ldots$$

where each L_{ij} is a literal. An atomic sentence is neither a universal clause nor an existential disjunction though a disjunction with a single disjunct is permitted. This seemingly-trivial insistence that every existential disjunction and universal clause contains a disjunction has the crucial consequence that these sentences are interpreted more weakly

by R than by T.

Property 1

There is a procedure, which given any finite set of univers clauses--kb--and any existential disjunction--q--decides whet $kb \models_R q$.

Though this crucial property is not proved here, the remaining prope ties mentioned in this paper are the key lemmas in its proof.

One intuition about retrieval is that it yields exact answers. example, "(E x) Liar(x)" is not retrievable from a KB containing sol "Liar(Richard-Nixon) v Liar(John-Dean)" because a particular x cannot named. So a retriever only says that an individual with a particu property exists if it is able to name that individual. The follow property formally states that $\models_R$ captures this intuition.

Property 2

An existential conjunction,

$$(E\ x_1\ x_2\ \ldots)\ \ (L_{11}\ v\ L_{12}\ v\ \ldots)\ \&\ (L_{21}\ v\ L_{22}\ v\ \ldots)\ \&\ \ldots$$

is R-logically implied by a set of universal clauses, kb, if, only if, there is a substitution, θ, of ground terms for the va ables $x_1, x_2, \ldots$ such that

$$kb \models_R (L_{11}\ v\ L_{12}\ v\ \ldots)_\theta \text{ and}$$
$$kb \models_R (L_{21}\ v\ L_{22}\ v\ \ldots)_\theta \text{ and } \ldots$$

Now consider the problem of retrieving a ground clause--a disjunction variable-free literals--from a set of universal clauses.

Property 3

Let kb be a set of universal clauses and q be a ground claus Then $kb \models_R q$ if, and only if, $b \models_R q$ for some b that is a gro instance of a clause in kb.

This is a no-chaining property where a fact is formalized as being ground clause. Notice that R-logical implication is restricted not j

to a single universal clause but to a single instance of the clause. This means that

$$\forall x \sim N(x) \vee N(s(x)) \not\models_R \sim N(0) \vee N(s(s(0)))$$

because the query T-logically follows only from two instances of the clause in the KB:

$$\sim N(0) \vee N(s(0)),\ \sim N(s(0)) \vee N(s(s(0))) \models_T \sim N(0) \vee N(s(s(0)))$$

Such an inference can be seen as chaining a sentence with itself, and hence our intuitions say that it should not be performed by the retriever. Property 3 confirms that the specified retriever captures this intuition.

There are many conceivable retrievers that, like this retriever, satisfy the no-chaining restriction for the definition of "fact" used here. However, among these retrievers the one specified here occupies a privileged position by virtue of Property 4.

Property 4

For any two ground clauses, kb and q, $kb \models_R q$ if, and only if, $kb \models_T q$.

Recall (from the soundness requirement of Section 4.1) that for a relation to be considered a retrievability relation it must be a subset of $\models_T$. Therefore, with respect to the present definition of "fact," $\models_R$ is the strongest retrievability relation satisfying the no-chaining restriction!

4.4. Extensions

I expect that having a model-theoretic specification will result in big payoffs when the capabilities of the simple retriever are extended. Consider extending the retriever to do a specific kind of finite chaining--for example, reasoning about inheritance and taxonomies as is typically done by semantic-networks. FOPC can be enhanced with notational devices for expressing information about taxonomies and the

Tarskian model theory can also be extended to deal with these syn additions. Our clear and simple intuitions about taxonomies mak an easy task. Since the taxonomic extensions are decidable an retriever is to reason fully about them, these same extensions made to R in order to obtain a retrievability relation for the en language.

Though the difficult task of finding a proof theory or a decision cedure still remains, imagine the difficulty of doing so witho guidance of a model theory. How would one know when all nec inferences were captured, or if the captured ones were reasonabl course one's intuitions could provide guidance, but not to the provided by a model theory constructed from the same intuitions.

In another application, R could be extended to form a logic of Levesque (1984) calls "explicit belief"--those beliefs that an can readily access (i.e., retrieve). R provides a useful foundat which to build since it accounts for certain crucial facts: an a explicit beliefs may be T-inconsistent, an agent cannot expl believe most of the T-consequences of his explicit beliefs, and an can explicitly believe that P but not that Q, even if P and Q a logically equivalent and he explicitly believes so.

However, R needs to be extended in order to account for certain a of explicit belief not present in retrieval. For example, while i fices to specify a retriever by associating its retrievability re with the logic's meta-theoretic logical-implication relation are many reasons why it is necessary to associate a belief re with a relation in the logic itself, not the least of which i beliefs can embed other beliefs.

5. Related Work

There is one line of research that has resulted in a logic so clos that I discuss it here at the exclusion of all else. The compari brief, though a detailed one certainly is called for and easily merit an entire paper.

Belnap (1975; 1977) presents a four-valued relevance logic that provides a weakened interpretation of propositional logic. Levesque (1984) in turn uses this logic as the foundation of a modal explicit-belief operator in a propositional logic. With the retrieval problem in mind, Patel-Schneider (1985) extends Belnap's logic with quantification, resulting in a system with a t-entailment relation strikingly similar to $|=_R$. Though not explicitly discussed, each of these systems provides a standard logical system with an alternative, relaxed model theory and in each case the motivation is to obtain a weak logical-implication relation (called "entailment" in these systems) with certain properties.

Propositional sentences in Patel-Schneider's system and those embedded in Levesque's implicit-belief operator have the same interpretation as in the underlying system of Belnap. Hence, I only shall compare propositional interpretations in R with those in Belnap's four-valued logic, which henceforth is called simply "B". Whereas the elimination of chaining motivated the development of R, the elimination of inconsistency motivated the development of B. To compare R and B directly, one can consider the four values that a B-model can assign to a sentence as corresponding to the four ways that an R-model can classify a sentence with respect to a designated world, w: True in w and True in no other world, True in w and True in some other world, False in w and True in no other world, and False in w and True in some other world. It then becomes clear that the two logics weaken the logical connectives of FOPC differently. R weakens the interpretation of disjunction so that a disjunction of false formulae may be either true or false. B weakens the interpretation of negation so that the truth of a sentence is not related to that of its negation--intuitively what one would expect from a logic designed to eliminate inconsistency.

The remarkable result is that, like R, B does not sanction the chaining of facts as defined here. However, B is weaker than R in that no sentence is B-valid. Whereas B denies the existence of validity, R carries the no-chaining intuition through to its definition: a normal form sentence is R-valid if, and only if, each of its facts is valid. every KB. Specifically, the relationship between the two systems is that for sentences in prenex conjunctive normal form

kb $|=_R$ q if, and onJy if, q is R-valid or

kb $1=_B$ q

6. Conclusion

Different specifications of a program provide different viewpoints of what a program does for a user. When the equivalence reasoner is queried "r(a,b)^M or when the retriever is queried "UNCLE(JOHN,RILL)," exactly what is being asked? According to a proof-theoretic specification a query asks about the existence of a proof, while according to a computational specification it asks for a certain computation to be performed. Yet someone making such a query, or writing a program to do so, is likely to think not that his query makes either of these requests but rather that it asks about a state of affairs in the intended interpretation. The model theory provides this view; it says that if all the sentences in the KB are true in the intended interpretation and the retriever responds "yes" to "UNCLE(JOHN,BILL)" then John is Bill's uncle.[5] So according to the model-theoretic specification, the query is about the universe being represented in the KB, not about proofs or computations.

Mode]-theoretic specifications of AT programs are useful for several reasons. They are more abstract than LISP-code specifications yet formal enough to prove that a specified program has certain properties, such as those discussed in Section 4.3. It is hard to imagine a form of specification that moves further in the direction of saying what is computed without saying how. Perhaps this accounts for why the retrieval specification presented here is extremely short, though an efficient implementation would require a program of moderate size. In addition to being technically valuable, a model theory can be a useful heuristic tool, serving to sharpen and extend our intuitions. Though a proof-theoretic specification also can possess some of these properties,

---

[5] This statement holds regardless of whether the intended interpretation is in model theory R or T.

ideally one would like to have both forms of specification available; the more-appropriate specification can then be chosen for any given task.

This paper gives, I hope, strong evidence that a model-theoretic specification of a program is a valuable tool. Furthermore, it suggests how these specifications may be produced for incomplete-reasoning programs. Yet the utility of a methodology must be demonstrated by more than two examples, however persuasive they may be. The important question that remains unanswered is "What is the range of AI programs that can be specified naturally with a model theory?" I stress the word "naturally" because a cumbersome model-theoretic specification will not provide the benefits discussed above. To take an extreme case, it is easy to imagine that any syntactic specification could be transformed into a model-theoretic specification simply by bringing the syntactic objects into the model theory, but hard to imagine that anything would be gained by such contortions.

During a period when the retrieval specification used mechanisms other than possible worlds, I often felt that the specification was stretching the methodology to its limits, that slight extensions to the retriever would introduce immense complexity to the specification. My pessimism subsided with the introduction of possible worlds, and the potential for extension is now one of the strong points of the retrieval specification. The use of possible worlds is just one element in a large body of well-studied model-theoretic devices and logical implication is just one of a range of model-theoretic relations, which potentially could be used in the construction of specifications. On this rests the hope of using the proposed methodology to specify a wide range of programs. Further attempts to develop and use the technique are needed to see how far it can be extended before breaking down.

Acknowledgements

I Jong believed that a model-theoretic account of my retriever woi prove useful, but felt that it would be extremely difficult to obte and had no idea of how to approach the problem. I suspect that the are many AI researchers with similar beliefs about their programs ar like them, I thought that maybe someday in the distant future I woi attack the problem. All that changed in December 1982 when lengthy dj cussions with David A. McAllester convinced me that it was a problem could solve and led to several ultimately-correct suggestions f approaching the problem. The retrieval specification in this paper the result of a two-year process that produced numerous incorrect and/ over-complex specifications. I am most grateful to Dave, without whon would have slept better for two years but never would have written tl paper. I am also grateful to Peter Patel-Schneider, Chris Mellis Remko Scha and Aaron Sloman for their useful comments on various ve sions of the paper, to James Allen, Andy Haas and Pat Hayes with whom have had extensive discussions on model theory and knowledge retrieve and to Fran Evelyn for her eagle-eyed proofreading. This work has be supported partially by grant GR/D/16062 from the Science and Engineers Research Council.

References

Allen, J.F., A.M. Frisch and D.J. Litman, "ARGOT: The Rochester Dialogue System," Proceedings of the Second National Conference on Artificial Intelligence, August, 1982.

Belnap, N.D., "How a Computer Should Think," in Contemporary Aspects of Philosophy, Proceedings of the Oxford International Symposium, 1975.

Belnap, N.D., "A Useful Four-Valued Logic," in G. Epstein and J.M. Dunn (eds.), Modern Uses of Multiple-Valued Logic, Dordrecht: Reidel, 1977.

Fikes, R.E. and N.J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence 2, 189-208, 1971.

Frisch, A.M. and J.F. Allen, "Knowledge Retrieval as Limited Inference," in D.W. Loveland (ed.), Lecture Notes in Computer Science: 6th Conference on Automated Deduction. New York: Springer-Verlag, 1982. Also appears in "Knowledge Representation and Retrieval for Natural Language Processing," Technical Report No. 104, Computer Science Dept., Univ. of Rochester, December 1982.

Kripke, S.A., "Semantical Analysis of Modal Logic I, Normal Propositional Calculi," Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 9, 67-96, 1963.

Levesque, H.J., "A Logic of Implicit and Explicit Belief," Proceedings AAAI-84, August 1984. Revised version appears as Technical Report No. 32, Fairchild Laboratory for Artificial Intelligence, August 1984.

Patel-Schneider, P.F., "A Decidable First-Order Logic for Knowledge Representation," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, August, 1985.

Tarjan, R.E. and J. van Leeuwen, "Worst-Case Analysis of Set Uni
Algorithms," Journal of the A.C.M. **31**, 245-281, 1984.

Tarski, A., "Der Wahrheitsbegriff in den Formalisierten Sprachen," St
dia Philosophica 1, 261-405, 1935. Translated as "The Concept
Truth in Formalized Languages," in A. Tarski (ed.), Logic, Sema
tics, and Mathematics, Oxford: Clarendon Press, 1956.