Reset reproduction of CMU Computer Science report CMU-CS-83-124. Revised version published in *IEEE Computer*, Vol. 16, No. 10, pp. 119–123. Reprinted July 1994. Current address: MIT Laboratory for Computer Science, Cambridge, Massachusetts.

Admissible State Semantics for Representational Systems

Jon Doyle

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213
U. S. A.

Abstract: Several authors have proposed specifying semantics for representational systems by translating them into logic. Unfortunately, such translations often introduce unnecessary detail and complexity. We indicate how many kinds of informal semantics can be transformed directly into formal semantics of no greater complexity. The key to avoiding the difficulties of logical translations is to recognize the difference between internal and external meanings.

This paper will appear in *IEEE Computer*.

This research was supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Government of the United States of America.

The Problem of Practical Semantics

§**1.** Although the design of representational systems involves many considerations, such as computational efficiency of the operations and accommodation (at least through front-ends) of the conceptual scheme to those familiar to humans, one of the most important requirements is for a clear semantics. Just as a slow program may be useless, and just as sermons in one church may bewilder the members of another, one cannot hope for success in representing information about the world if one cannot tell what the representations mean. Without a clear semantics, one cannot tell if two representations mean the same thing or mean different things. This prevents judging the correctness of formalizations of one's intuitive knowledge. Similarly, without a clear semantics, one cannot distinguish innocuous, meaning-preserving inferences from inferences which introduce new assumptions or change meanings.

Artificial intelligence has pursued two paths towards formalizing the semantics of representational systems. Both of these are based on mathematical logic. In one, the representations themselves are sentences in a logical language, as in PROLOG; in the other, one gives a translation of every representation into a set of sentences in a logical language, as illustrated by [Hayes 1979] and [Nilsson 1980]. In either case, the meaning of representations is found by looking at the models of the corresponding set of sentences.

Unfortunately, neither of these paths offers any guarantee that the resulting semantics will be easy to construct or to comprehend. The source of the difficulty is that many sorts of representations important in artificial intelligence concern self-knowledge of one kind or other, representations of the agent about its own structure and behavior. Some of these self-representations are purely descriptive or introspective. Some are used normatively as ideal "self-images" to uphold in actual thought and action. Phrasing such self-representations or their translations in a logical language is not impossible, but usually requires very complex constructions involving convoluted language-metalanguage systems. This is not theoretically objectionable, but it spells trouble in practice, since extremely complex translations are difficult to comprehend by the designer and user, making translation-based semantics ill-suited to its principal mission. This problem shows up often in artificial intelligence, where many representational systems are described in non-linguistic or non-logical terms and never supplied with formal meanings, even when reasonably clear informal semantics are obvious.

In the following, we indicate how many kinds of informal semantics can be transformed directly into formal semantics of no greater complexity. We avoid the unnecessary burdens of logical formulation and translation by focusing on what is real — the meanings — rather than on their expression within a particular logical language. Translations into logical languages, while theoretically sufficient, are not unique, since many languages can serve if even one can. Yet each logical language introduces peculiarities of its own, details that impede understanding and analysis without affecting the resulting meanings.

The key to avoiding the difficulties of direct logical translation is to recognize the difference between the internal and external meanings of representations.

Internal and External Meanings

§**2.** Designers of artificial intelligence systems commonly employ two sorts of reference: *external reference*, in which the agent's representations refer (in the designer's mind or by some other means) to objects not immediately "graspable," for example diseases and geological formations; and *internal reference*, in which the agent's representations refer (ostensively or otherwise) to objects immediately graspable and hence in the agent itself. This is reflected in the common focus of artificial intelligence architectures on general manipulations of representations rather than on pure logical deductions. If one has immediate access

to an object, one can not only talk about it, but modify it. With immediate access, if one makes an inaccurate statement, one can either retract it or make it true — both important operations in artificial intelligence systems. On the other hand, without graspability, one can only talk about objects, and cannot quickly modify them to cover one's assertions. This makes deduction an important way of talking about ungraspable objects, since one can say new things without fear of being more wrong than before. The principal novelty of current representational systems relative to traditional systems of deductive logic is their concentration on the use of the relatively neglected tools of internal reference as the basis of self-structuring and self-modifying agents, agents which can state their own intended structure, and then make those statements true if need be.

PUTNAM [1975] and others argue that even supposedly mental objects like human beliefs cannot actually be grasped, and so raise doubts that two sorts of reference exist, doubts that anything can be immediately grasped at all. The artificial intelligence approach is based on ensuring that some objects are actually grasped by construction. While many representations of the usual knowledge representation systems have external referents, and so do not directly affect the mind (except through mistakes that cause injury or death), many of the information-structuring representations concern mental objects themselves, especially relations between representations. Rather than ask PUTNAM'S question of whether these structural representations *actually* mean what the agent thinks they do, we use the intra-mental relations to *define* the *admissible states* of the agent. The problem of correct implementation of these self-representational specifications is that of implementing the agent so that the states of the implementation are exactly the admissible states, that is, so that the structural representations have exactly their intended meaning. We separate this portion of the meaning of mental components from general ecological meaning by the name *admissible state semantics*, and leave specification of external meanings to the standard tools of model theory.

The method of admissible state semantics is simple, and resembles the usual explanations of intended meanings given by system designers. In both of these, the designer explains the meaning of one representation directly in terms of its relations to other representations in the system. For comparison, the logic translation approach requires one first translate the initial representation into logic, then find its consequences, and then reverse the translation process to find other representations related to the original one. Since many representational systems involve succinct encodings of notions whose logical translations are very complex, the difficulty of this roundabout logical procedure can be unbearable.

Admissible State Semantics

§**3.** Admissible state semantics makes several fairly general assumptions about the constitution of agents. These constitutive assumptions involve some "parameters," so one applies the framework by filling in these parameters with the characteristics intended of one's system. The three fundamental parameters are called $\mathcal{D}$, $\mathcal{I}$, and $\mathcal{S}$. We explain these in turn. (These three notions are part of a larger framework developed in [DOYLE 1982] and elsewhere.)

The first constitutive assumption is that every state of the agent can be decomposed into elements drawn from a domain $\mathcal{D}$. Here $\mathcal{D}$ is just a set, so each state S is a subset of $\mathcal{D}$, that is, $S \subseteq \mathcal{D}$. For most purposes, $\mathcal{D}$ is just the set of all possible representations or data-structures the system might employ. For example, a logically-structured agent might be characterized by taking $\mathcal{D}$ to be the set of all sentences in some logical language; LISP-based agents might require $\mathcal{D}$ be the set of all possible S-expressions; for frame or unit-structured agents, $\mathcal{D}$ can be the set of all possible frames or units; semantic networks likely require $\mathcal{D}$ be the set of all possible nodes and links; and "society of mind" agents can be described using $\mathcal{D}$ as the set of all "mental agents" (see [DOYLE 1983]). Note carefully that $\mathcal{D}$ is not just the set of all components in some particular state, such as the initial state, but instead the set of all components that might appear in any state, at any time. It is possible, without much trouble, to formalize one's system

instead in terms of an increasing sequence of domains (to capture "generated symbols" or other additions), but illustrating that would digress too far from our main purpose here.

The second constitutive assumption is that every element of the domain, every possible state component, represents a specification on the set of states in which it may admissibly occur, and has a meaning or interpretation that sets out these sanctioned states. Formally, we assume an interpretation function $\mathcal{I} : \mathcal{D} \rightarrow \mathbf{P}\,\mathbf{P}\,\mathcal{D}$ ($\mathbf{P}$ means power set), so that for each $d \in \mathcal{D}$, $\mathcal{I}(d) \subseteq \mathbf{P}\,\mathcal{D}$ is the set of potential states sanctioned by d. For example, state components that are indifferent to the states in which they appear (such as representations purely about the external world) can be given the trivial interpretation $\mathcal{I}(d) = \mathbf{P}\,\mathcal{D}$ that sanctions all potential states. If the component requires that its appearance always be accompanied by some other components $A \subseteq \mathcal{D}$, then one can define $\mathcal{I}(d) = \{S \subseteq \mathcal{D} \mid A \subseteq S\}$. To forbid the component from occurring with some other components $A \subseteq \mathcal{D}$, we can define $\mathcal{I}(d) = \{S \subseteq \mathcal{D} \mid S \cap A = \emptyset\}$. Of course, these are very simple sorts of interpretations. Sophisticated systems may have some components that play very involved roles in the agent, and these may require very complex interpretations. Note carefully that one is free to use whatever precise (e.g. mathematical or logical) language is convenient in defining the interpretations of components. These metalanguages are part of *our* (external) specification of the system, and need have no close relation to the system's own methods of representation. Put another way, we can use logic to characterize the intended behavior of the agent without having to pretend the agent's components and actions are logical sentences and logical inferences. This, as I see it, is the principal advantage of the proposed semantical framework over those based on logical translations. To ease the semanticist's burden even more, we note that when there are several overlapping classes of components with special interpretations, one can specify several separate interpretation functions, one for each class of components, and then intersect them to get the full interpretation function. For example, if $A, B \subseteq \mathcal{D}$ each contain related sorts of components, one can define for every $d \in \mathcal{D}$

$$\mathcal{I}_A(d) = \begin{cases} \ldots & \text{if } d \in A \\ \mathbf{P}\,\mathcal{D} & \text{otherwise} \end{cases}$$

$$\mathcal{I}_B(d) = \begin{cases} \ldots & \text{if } d \in B \\ \mathbf{P}\,\mathcal{D} & \text{otherwise} \end{cases}$$

$$\mathcal{I}(d) = \mathcal{I}_A(d) \cap \mathcal{I}_B(d).$$

The third constitutive assumption is that every *admissible state* of the system satisfies the specifications represented by each of its components. We write $\mathcal{S}$ to mean the set of admissible states of the agent, and define the class $\mathcal{Q}$ of *component-admissible sets* by

$$\mathcal{Q} = \{S \subseteq \mathcal{D} \mid S \in \bigcap_{d \in S} \mathcal{I}(d)\},$$

so this constitutive assumption is that $\mathcal{S} \subseteq \mathcal{Q}$. Unfortunately, simple component-admissibility cannot capture some intended ranges of admissible states for agents. For example, the empty set $\emptyset$ is always component-admissible since it has no elements to say otherwise. One might wish to capture other restrictions on the intended states without explicitly representing them by interpretations of components. To allow this, the framework permits definition of $\mathcal{S}$ as a proper subset of $\mathcal{Q}$. Note that we can always capture any general restriction except non-emptiness in the components themselves by redefining $\mathcal{I}(d)$ as $\mathcal{I}'(d) = \mathcal{S}$ for every $d \in \mathcal{D}$, in which case $\mathcal{Q}' = \mathcal{S} \cup \{\emptyset\}$. Turning this observation around, if $\mathcal{S} = \mathcal{Q}$, then all restrictions on states are explicitly represented in the states themselves. This recalls current efforts in artificial intelligence aimed at constructing completely "self-descriptive" systems, but we cannot pursue those here.

In the following examples, we present some semantical specifications using this framework. Unfortunately, demands for brevity limit what we can present here. More detailed and comprehensive treatments of major artificial intelligence systems are in preparation.

§**4.** Many systems represent information in so-called semantic networks. One of the fundamental sorts of information encoded in these representational systems concerns the "inheritance" of information by one concept from another. If we look for logical translations of these systems, the temptation is strong to formulate inheritance as implication, since everything derived about one concept can be derived in one further step about any concept it implies. Unfortunately for the simplicity of logical translations, many uses of inheritance in artificial intelligence have nothing to do with implication, but instead concern simple economy in writing down information. One often sets up inheritance relations not to indicate any common referents of descriptions, but as cheap ways of constructing one description in terms of its differences, both positive and negative, from another. For example, if we already have a description of lions, we can quickly construct a description of tigers by declaring that the two descriptions are the same, except (say) that tigers look different and live in India. Here we save rewriting all the information about being mammals, quadrupeds, furred, and so on, yet do not state that all tigers are lions, nor even that some tigers are lions. We just say "ditto." Considerable investigation still continues on what notions of inheritance are practically useful and theoretically important in general and in specific cases. We add nothing to those debates here, but instead illustrate how the semantical framework introduced above allows designers of such systems to state their intended conceptions of inheritance exactly and independently of how they implement those conceptions.

To give perhaps the most trivial example possible, suppose we choose to represent concepts by LISP atomic symbols with property lists, and intend that any concept with an **IS-A** property should also have every property of the concepts listed under the **IS-A** property. Formally, we let $\mathcal{D}$ be the set of all LISP S-expressions, take $\mathcal{S} = \mathcal{Q}$, and define $\mathcal{I}$ so that $\mathcal{I}(d) = \mathbf{P}\,\mathcal{D}$ if d is not an atomic symbol. We write $p(a) = x$ to mean that the atomic symbol a has x as the value of its p property. With this notation, we specify the interpretation of each atomic symbol a by

$$\mathcal{I}(a) = \{S \subseteq \mathcal{D} \mid \forall b \in \textbf{IS-A}(a) \quad \forall p \neq \textbf{IS-A} \quad p(b) \neq \mathit{NIL} \supset p(b) = p(a)\}.$$

That is, except for the **IS-A** property itself, which must be treated differently in this representation, the inheriting concept must have all the properties of its ancestor. Since the ancestor imposes similar conditions on states, inheritance is "transitive" in every admissible state. Of course, no one would ever want to use such a simple-minded system: its limitations are obvious. But we can extend the same methods to more interesting representational systems.

Consider SRL, the "schema representation language" of Wright and Fox [1982]. One important feature of SRL is the definability of special classes of inheritance types within the representation system itself. While the full language is too large to present here, we can focus on one typical feature which illustrates how one might begin to formally specify the internal semantics of all of SRL. For this fragmentary analysis, we take $\mathcal{D}$ to be the set of all possible SRL "schema" data-structures. The precise extent of this set does not matter for this example. Indeed, we use little more than the resemblance of schema to the simpler property-list data-structures discussed above, and so do not do justice to SRL proper. The focus of our attention is the "inclusion-spec inheritance schema." An inheritance schema describes a class of inheritance relationships, and inclusion-specs are generalizations of **IS-A** relationships. Wright and Fox display the form of inclusion-spec schemata as

{{ **inclusion-spec**

DOMAIN:	$<$ *restriction* $>$
	default: all
RANGE:	$<$ *restriction* $>$
	default: all
TYPE:	
	default: value
	range: (SET (OR slot value))
SLOT:	$<$ *restriction* $>$
	default: all
VALUE:	$<$ *restriction* $>$
	default: all
CONDITION:	
	default: T
	restriction: (OR $T <$ *predicate* $>$) }}

They explain this schema as follows. Every inheritance schema has two slots called SCHEMA1 and SCHEMA2 whose fillers are the schemata to be related by the defined inheritance relationship. The inclusion-spec has a number of additional slots which, taken together, describe exactly which sorts of information should be transferred from SCHEMA1 to SCHEMA2. DOMAIN and RANGE may be filled with predicates on schemata limiting the force of the inclusion-spec to pairs of schemata satisfying the respective restrictions. CONDITION is in addition a general predicate that must be satisfied for the inclusion-spec to transfer information. The TYPE slot indicates whether only slots, or slots and their values are to be transferred. The SLOT slot allows transfers to be restricted to a subset of SCHEMA1's slots, and the VALUE slot can restrict the sorts of values passed to SCHEMA2. In this way, the inheritance schema encodes a general statement about information transfer in the agent, and one uses the schema by filling in the ranges of some of the implicit quantifiers and referents of some of the explicit names. Use of a schema to state this specification rather than an arbitrary sentence of logic implicitly limits the user to specifications which the system implementor decides can be feasibly computed. To formally specify the semantics of SRL with just this one sort of relation schema, we need only define $\mathcal{I}(d) = \mathbf{P}\,\mathcal{D}$ when d is anything other than an inclusion-spec, and define the cases for an inclusion-spec d analogously to the property-list example above, perhaps by

$$
\begin{aligned}
\mathcal{I}(d) = \{S \subseteq \mathcal{D} \quad | \quad & \mathit{apply}(d.\mathrm{DOMAIN}(S), d.\mathrm{SCHEMA1}(S)) = T \\
& \wedge \mathit{apply}(d.\mathrm{RANGE}(S), d.\mathrm{SCHEMA2}(S)) = T \\
& \wedge \mathit{eval}(d.\mathrm{CONDITION}(S)) = T \\
& \wedge \forall s \in d.\mathrm{SCHEMA1}.\mathit{slots}(S) \\
& \quad [\mathit{apply}(d.\mathrm{SLOT}(S), s) = T \supset s \in d.\mathrm{SCHEMA2}.\mathit{slots}(S) \\
& \quad \wedge d.\mathrm{TYPE}(S) = \mathrm{value} \supset \\
& \quad \forall v \in d.\mathrm{SCHEMA1}.s.\mathit{values}(S) \\
& \quad [\mathit{apply}(d.\mathrm{VALUE}(S), v) \supset v \in d.\mathrm{SCHEMA2}.s.\mathit{values}(S)]]\}.
\end{aligned}
$$

By doing enough honest work in defining these subsidiary functions (which we cannot pretend to here), we can continue in this way to give meanings to other sorts of SRL schemata as well.

§**5.** To illustrate the applicability of admissible state semantics to non-linguistic structures for agents, we consider some elements of MINSKY'S [1980] K-line theory of memory. For MINSKY, the mind is composed of a set of "mental agents." Each mental agent can be either active or inactive, and states of mind are simply sets of active mental agents. We identify the set of mental agents with the domain $\mathcal{D}$ of the agent, and consider sets in $\mathcal{Q}$ to be the admissible sets of active mental agents, that is $\mathcal{S} = \mathcal{Q}$.

The two specific sorts of mental agents we formalize here are K-lines and cross-exclusion networks. K-lines are mental agents that, when activated, cause the activation of some set of other mental agents. We formalize this by interpreting each K-line mental agent KL in terms of the set A of mental agents to which it is connected, so that

$$\mathcal{I}(KL) = \{S \subseteq \mathcal{D} \mid A \subseteq S\}.$$

Cross-exclusion networks are somewhat more complicated. Cross-exclusion networks are sets of mental agents which are mutually inhibitory. Further, cross-exclusion networks facilitate "conflict resolution" by disabling or ignoring all members if two or more manage to become active despite their mutual inhibitions. This disabling allows activation of "higher-level" mental agents which can consider and resolve the conflict. We might formalize this by letting CXN be a mental agent representing a cross-exclusion network, $B = \{b_i\}_{i=1}^n$ the set of mutually inhibiting members, $C = \{c_i\}_{i=1}^n$ indicators of which competitor wins out, and $\overline{CXN}$ a mental agent representing the existence of an externally forced conflict. To get the desired behavior, we define

$$\mathcal{I}(CXN) = \{S \subseteq \mathcal{D} \mid [\overline{CXN} \notin S] \supset B \subseteq S\},$$

$$\mathcal{I}(b_i) = \{S \subseteq \mathcal{D} \mid [S \cap (C - \{c_i\}) = \emptyset] \supset c_i \in S\}$$

for each i, and assume the existence of a "watchdog" WD such that

$$\mathcal{I}(WD) = \{S \subseteq \mathcal{D} \mid [\exists i \neq j \leq n \quad c_i, c_j \in S] \supset \overline{CXN} \in S\}.$$

With these interpretations, we can capture the meanings or functions of mental agents without having to dissect them.

Conclusion

§**6.** We have indicated the internal meanings of a variety of representational systems without translating representations into a logical language. Unfortunately, demands for brevity limit the scope of this paper, and we have had to omit treatment of virtual state information and action specifications. But as a final remark, we note that the proposed semantical framework provides a starting point for investigating SMITH'S [1982] representational hypothesis. According to SMITH, many workers in artificial intelligence suppose that in any "interesting" computational agent, the representational elements of the agent's structure can be viewed propositionally, and that the computations made by the agent depend purely on the form, not on the content, of these elements. Such propositional perspectives on the structure of agents may seem quite elusive if we look at non-linguistic structures like K-lines, but the preceding framework offers tools for reconstructing the propositional structure of non-linguistic agents. For example, if $\mathcal{I}(e) = \mathcal{I}(e_1) \cap \mathcal{I}(e_2)$, one might think of e as the statement "$e_1 \wedge e_2$." Similarly, if $\mathcal{I}$ is defined as the intersection of several restricted interpretation functions $\mathcal{I}_1, \ldots, \mathcal{I}_n$, one can view the domains of nontriviality of these functions as the "syntactic classes" of the agent's language. Even so, one may not find this reconstructed "language" looking anything like a full first-order logical language. What the current framework suggests, I think, is that one can make sense of the formality condition of the representational hypothesis without worrying too much about linguistic re-representability of the agent's structure. This brings us back to the initial proposal of this paper, that there are easier ways to give exact semantics to representational systems than translations into logic.

REFERENCES

Doyle, J., 1982. Some theories of reasoned assumptions: an essay in rational psychology, Pittsburgh: Department of Computer Science, Carnegie-Mellon University.

Doyle, J., 1983. A society of mind: multiple perspectives, reasoned assumptions, and virtual copies, *Eighth International Joint Conference on Artificial Intelligence*.

Hayes, P. J., 1979. The logic of frames, *Readings in Artificial Intelligence* (B. L. Webber and N. J. Nilsson, eds.), Palo Alto: Tioga, 451-458.

Minsky, M., 1980. K-lines: a theory of memory, *Cognitive Science* **4**, 117-133.

Nilsson, N. J., 1980. *Principles of Artificial Intelligence*, Palo Alto: Tioga. Chapter 9.

Putnam, H., 1975. The meaning of 'meaning,' *Mind, Language, and Reality*, Cambridge: Cambridge University Press, 215-271.

Smith, B. C., 1982. Reflection and semantics in a procedural language, Cambridge: Laboratory for Computer Science, Massachusetts Institute of Technology, TR-272.

Wright, M., and Fox, M. S., 1982. SRL/1.5 user manual, Pittsburgh: Robotics Institute, Carnegie-Mellon University.