A First Order Theory of Planning, Knowledge, and Action

Leora Morgenstern

New York University

Department of Computer Science

New York, N.Y. 10012

ABSTRACT

Most AI planners work on the assumption that they have complete knowledge of their problem domain and situation so that planning an action consists of searching for an action sequence that achieves some desired goal. In actual planning situations, we rarely know enough to map out a detailed plan of action when we start out. Instead, we initially draw up a sketchy plan and fill in details as we proceed. This paper presents a formalism based upon a syntactic logic of knowledge which is expressive enough to describe this flexible planning process.

1. Introduction

Most AI planners work on the assumption that they have complete knowledge of their problem domain and situation, so that planning an action consists of searching through a pre-packaged list of action operators for an action sequence that achieves some desired goal. Real life planning rarely works this way, because we usually don't have enough information to map out a detailed plan of action when we start out. Instead we initially draw up a sketchy plan and fill in details as we gain more exact information about the world.

A robust planning system must have similar capabilities if it is to work in a complex domain. In particular, it needs a solid theory of knowledge, action, and communication, so that it can produce a coherent plan even when its knowledge of the problem domain and problem situation is incomplete. Some work has already been done in this field, most notably by Robert Moore [Moore 1980], who has developed a theory of knowledge and action in which it is possible, for example, for an agent who knows the combination of a safe to reason that he knows how to open the safe. Due to the inherent limitations of Moore's formalism, however, Moore cannot attack the problems of knowledge, action, and communication in their full generality.

This paper presents a logic that is expressive enough to deal with the problems that faced Moore's system. In the next section of this paper, we review Moore's work on knowledge and action and demonstrate its logical limitations. Afterwards, we present an alternate approach, discuss its power, and show how it can be integrated with Moore's work on knowledge and action. Finally, we show how this logic of knowledge and action can be extended to a more robust theory of planning, and demonstrate solutions to a number of

problems that cannot be handled by current theories of knowledge and action.

2. Moore's Logic of Knowledge and Action

The bulk of Moore's work on knowledge and action is directed towards explaining how knowledge affects action. Prior to Moore's pioneering work, researchers such as McCarthy and Hayes [1969] had argued that a planning program needs to explicitly reason about its ability to perform an action. These researchers suggested writing down explicit knowledge precondition axioms for each action, so that a planning program could reason that it knew how to do an action if the relevant knowledge precondition axioms were true. Unfortunately, this approach leads to an explosion of knowledge precondition axioms and unacceptably long proofs. This problem, which we will call the problem of knowledge preconditions, is the one that Moore's system seeks to solve.

Moore uses a standard S4 modal logic of knowledge, with the following axiom schemata:

M1: Axioms of ordinary propositional logic
M2: Know(A,P) => P (veridicality)
M3: Know(A,P) => Know(A,Know(A,P)) (positive introspection)
M4: Know(A,P=>Q) => (Know(A,P) => Know(A,Q)) (consequential closure)
M5: If P is an axiom, then Know(A,P) is an axiom (necessitation)

The semantics for 'know' are given in terms of a Kripkean possible worlds semantics [Kripke 1963a&b, Hintikka 1962 & 1969]: an agent knows p in a particular world if and only if p is true in all worlds that are knowledge equivalent to his world. By positing that 'know' is a reflexive and transitive relation, we get the effect of M1-M5.

The notion of a rigid designator - something that denotes the same individual in all possible worlds [Kripke 1972] - plays an important role in Moore's theory of knowledge and action. According to Moore, we can in general say that we know who somebody is or what something is if we know a rigid designator for that person or object. Similarly, says Moore, we know what an action description is, and thus how to do the action, if we know of a rigid designator for that action. Moreover, we know how to do an action if we know of an

executable description of that action. Moore thus argues that a rigid designator for an action must be its executable description.

In practice, continues Moore, most action types are rigid functions which map rigid designators onto rigid designators, or are axiomatically defined in terms of other rigid functions. Thus if an agent knows rigid designators for the parameters of such an action, he knows a rigid designator for the action itself, and thus knows how to do the action.

Moore has thus characterized the knowledge preconditions for all actions with one principle. He has effectively dealt with the problem of knowledge preconditions while presenting a cogent explanation of what it means for an agent to know how to do an action.

2.1. Criticisms of Moore's System

Unfortunately, Moore's system is beset by a host of problems. In the first place, it is intuitively implausible that an executable description of an action can serve as a rigid designator for that action. Rather, an executable description of an action seems to be a property of that action that varies among different possible worlds. For example, in 1950 the executable description of dial(911) was a sequence of rotate-dial movements, while in 2050, if dial phones will be obsolete, the executable description of that same action might be a sequence of push-button movements. Given any executable description of an action, we can imagine some possible world in which that executable description would not work.

Secondly, Moore's assumptions regarding an agent's knowledge seem excessively strong. In his system all agents know all axioms. This implies that all agents have the same level of procedural knowledge, so that all an agent might not know is some of the slot fillers in some action. Yet there are clearly many situations in which an agent cannot do an action because he has no idea how to do the general procedure. Moore's system admits of no such possibility, since all general procedures for action are known. Moreover, it is impossible in Moore's system to relax the requirement that all axioms are known by all agents; this is a direct consequence of the possible worlds semantics on 'know.'

Thirdly, Moore's system is simply not expressive enough to handle a lot of our statements about knowledge. This criticism in fact applies to any standard first order modal logic of knowledge. Because we cannot quantify over sentences, we cannot formulate such sentences as 'John knows that Bill knows something that he doesn't know.' Assuming that knowledge about actions is in the form of statements, we also cannot express 'John knows that Bill knows how to fire a gun' unless John himself knows how to fire a gun.

Finally, we note that Moore's system deals with only one of the many issues that must be addressed by a complete theory of knowledge and action. An agent must not only be able to reason about whether he knows how to do an action at a particular time, but should also be able to reason about whether he might eventually know how to do that action, how he can learn to do the action, and whether he can delegate the action to some subordinate agent.

3. An Alternate Approach: 'Know' as a Syntactic Predicate

Our first task in developing a theory of knowledge and action is choosing a formalism for our theory. We reject first order modal logic, since that turns out to be insufficiently expressive for our needs. A move to a higher order modal logic [e.g., Gallin 1975] is likewise rejected because we would no longer have a complete proof procedure, and a higher order modal logic with possible worlds semantics would still entail that all agents know all axioms. Instead, we turn to a first order predicate logic with quotation, where 'know' is a syntactic predicate that ranges over names of sentences. Such an approach allows us to formulate the examples of the last section that gave Moore's system so much trouble. For example, 'John knows that Bill knows something that he doesn't know' can be formulated as Know(John,'Exists x (Know(Bill,x) and ~Know(John,x))'). Although we are effectively quantifying over sentences, our system remains first order because we are really quantifying over *names* of sentences, which are just strings or numbers.

Unfortunately, treating 'know' as a syntactic predicate leads to severe difficulties [Montague 1963]. In their simplest form, these difficulties manifest themselves as the

Knower Paradox [Montague and Kaplan 1960]: If 'know' is a syntactic predicate, we can construct a sentence S such that S iff Know(a,'~S'). Assuming veridicality, consequential closure, and necessitation on these two principles, S is inconsistent.

Providing some resolution to the Knower Paradox is thus a prerequisite to any syntactic theory of knowledge. We first observe that the Knower Paradox is just a variant of the Liar Paradox: if we allow 'true' to be a syntactic predicate in a classical logical language, we can construct a sentence such as P iff ~True('P'), which is of course inconsistent. To avoid this paradox, Tarski suggested that we have a hierarchy of languages and a hierarchy of truth predicates, one truth predicate per language. In no case is a language allowed to contain its own truth predicate, so we cannot construct paradoxical statements. Konolige [1981] has suggested constructing a hierarchy of know predicates to avoid the Knower Paradox, but we reject this approach because of various unsatisfying features of Tarski's hierarchy of predicates [Kripke 1975]: In the first place, it is implausible that people are consciously aware of using different truth predicates when they speak. Secondly, we often do not know which truth predicate to use when we utter a particular statement. Finally, it is impossible, in Tarski's approach to construct a pair of sentences, each of which refers to the truth value of the other, although such sentences may make perfect sense. Suppose, for example, that John Dean says 'All of Nixon's utterances about Watergate are false' and that Nixon says 'Everything Dean says about Watergate is false.' If both Dean and Nixon have made some trivial but true statement about Watergate, both statements will be false, but will certainly make sense. Moreover, the truth values of statements such as these often depend on empirical facts about the world, not on the syntactic structure of the statements.

These problems have led Kripke [1975] to develop a theory of truth which avoids both the Liar Paradox and the unattractive features of Tarski's approach. Our strategy will be to examine Kripke's construction and adapt it to a theory of knowledge which avoids the Knower Paradox.

Kripke's basic idea is that not every sentence is assigned a value of true or false;

viciously self-referential statements whether paradoxical or non-paradoxical get an indeterminate truth value, the third value in a three valued logic. Kripke's three valued logic is based on Kleene's [1952]: a or b is true if a is true or b is true, false if both a and b are false, of indeterminate value otherwise; forall x P(x) is true if P(x) is true for all x, false if P is false for at least one x, and undetermined otherwise.

Like Tarski, Kripke considers a hierarchy of languages, but only one truth predicate T(x). He starts out with a language L_0 which contains only sentences that do not involve the concept of truth. The language at the next level L_1 contains all the sentences of L_0, plus those sentences which talk about the truth or falsity of sentences in L_0. This process continues as we ascend the hierarchy and is defined for every L_α where α is an ordinal. In general, at successor levels we take the truth predicates over the previous level; at limit levels we take the union of all sentences declared true or false at previous levels. At some level L_σ we will no longer be able to assign truth values to any more statements no matter how much higher we ascend the hierarchy. We call those statements that have been assigned a truth value in L_σ grounded; all other statements are called ungrounded and take on the third, indeterminate truth value of the three valued logic. Moreover, we can show that viciously self referential statements such as the Liar Sentence are ungrounded in such a construction.

3.1. Relevance of Kripke's Work to a Logic of Knowledge

Our strategy for avoiding the Knower Paradox will be to construct a language in which sentences like S iff Know(a,"S') as well as the Liar and Truthteller sentences are not grounded. Like Kripke, we will construct a hierarchy of languages, building up towards a language which contains its own truth and knowledge predicates. We will be explaining knowledge as true belief, though for the purposes of this construction we could as well have taken knowledge to be justified true belief; the former strategy is adopted purely for reasons of simplicity. We start off with a classical first order language L, which comes with a fixed set of predicates and relations, including the relation Believe(a,p). Believe(a,p) is a well

defined function on all strings p.[1] Our first language L_0 contains only sentences that do not involve the concepts of truth or knowledge. As we ascend the hierarchy, more and more of these sentences will get truth values. In general, at any (successor) level i, we say that Know(a,p) is true in L_i if p has a positive truth value in L_{i-1} and a believes p; Know(a,p) is false if p has a negative truth value in L_{i-1}, *or* if p is true in L_{i-1} but a does not believe p; and that Know(a,p) is undefined otherwise. T(x) and Know(a,x) have truth values at a limit level iff they have truth values at a lower level. We can show that this construction is monotonic: that as we ascend the hierarchy the extensions of T(x) and Know(a,x) increase, and that we eventually reach a fixed point L_σ such that the extension of T(x) and $\bigcup_a$ Know(a,x) remain the same for L_σ and $L_{\sigma+1}$. As before, we call a sentence grounded if it has been assigned a truth value at L_σ; otherwise it is ungrounded. It is easy to see that sentences like the Knower sentence as well as the Liar and Truthteller sentences are ungrounded in such a construction. We have thus successfully resolved the Knower Paradox and have dealt with the primary objection to a logic that treats 'know' as a syntactic predicate.[2]

4. Planning, Knowledge, and Action

We can now proceed to develop our theory of knowledge and action. In this section, we expand Moore's solution to the problem of knowledge preconditions and integrate it with a first order theory of knowledge and action. The resulting theory can handle Moore's benchmark problems: agents can reason that they know how to perform an action or sequence of actions, and they can obtain knowledge as the result of an action. We then extend our theory so that we can deal with more complex planning problems. We show that

1-To avoid the Believer Paradox that such a move entails, we reject the 'assumption of arrogance': Believe(a,'Believe(a,p) => p').

2-It should be noted that this logic is not classical. In particular the law of the excluded middle does not hold for ungrounded sentences. Thus the axiom schemata T('x') => x, x =>T('x'), Know(a,'x') => x do not hold for ungrounded sentences. We do however, have the classical inference rules T('x')/x, x/T('x'), Know(a,'x')/x, and can demonstrate that classical logic does hold for grounded statements. For attempts to resolve the paradoxes within classical two valued logic, see Perlis 1981, Gupta 1982, Herzberger 1982, Asher & Kamp, 1986. (These systems all lose the classical inference rules.)

we can deal with an agent who plans to learn how to cook by enrolling in cooking classes, and an agent who accomplishes a task that he is incapable of doing by assigning it to subordinate agents.

4.1. The Basic Theory

We start by giving a brief description of our language. We will be using a first order language L and a temporal logic which is loosely based on the work of McDermott [1982]. L comes with a set of (possibly partially) interpreted predicates, functions, constants, and variables. The distinguished variable s ranges over situations. A collection of consecutive situations is known as an interval. Actions and events are collections of intervals; an action and an agent map onto an event. All predicates are qualified by a situation or interval. Well formed formulas are built up in the usual way out of the basic building blocks of L using the standard logical connectives v and ~ and the universal quantifier forall. Given a wff x, we can apply the godelization function G to obtain the string that represents x. In general, we write G(x) as 'x'. G is invertible; given any string we can recover the formula it represents. Assuming a model M, we now give the semantics for True and Know:

(1) M |= T(p) iff M |= G^{-1}(p)

(2) M |= Know(a,p,s) iff M |= T(p) and M |= Believe(a,p,s)

Note that (1) implies that T(a v b) iff T(a) v T(b) and T(~a) iff ~T(a).

We note that the quoting mechanism is *opaque* in many respects so that saying things correctly becomes quite complicated. To facilitate writing down axioms, we introduce two syntactic abbreviations: a name-of operator @, where @ applied to an object yields the name of that object, and an antiquote operator ^ ^, where ^ ^ applied to a string variable p yields the string that p stands for.

We are now ready to present some axiom schemata that capture our basic intuitions on knowledge. Not surprisingly, these schemata are rather similar to Moore's M1-M5; whether we treat 'know' as a modal operator or as a syntactic predicate, our basic intuitions on

knowledge remain the same. It should be noted that the schemata below, particularly K2-K4, hold only for grounded statements.

K1: Axioms of ordinary predicate logic (see [Mendelson 1964], Section 2.3)
K2: Know(a,p,s) => p (veridicality)
K3: Know(a,p,s) => Know(a,'Know(@a,^p^,@s)',s)) (positive introspection)
K4: Agents know the rules of inference:
(i) Know(a,'implies(^p^,^q^)',s) and Know(a,p,s) => Know(a,q,s)
(Modus Ponens)
(ii) Know(a,'p',s) => Know(a,'Forall x (p)',s)
(Generalization)
K5a:If p is an axiom of predicate logic then Know(a,p,s) for any s
K5b:Know(a,K1-K5b,s) for any s

It will be noticed that K1-K5 differ from M1-M5 in two important ways:

(1) K1 and K4 are quite a bit stronger than their counterparts, M1-M4. M1 and M4 just specify that all axioms of propositional logic are in the system and that agents can and do use the rule of inference Modus Ponens. Thus, the system supports only inferences made via the rules of propositional logic, and assumes that agents are limited to these inferences as well. In contrast, K1 specifies that all axioms of predicate logic are in the system, and K4 posits that agents can and do reason with the rules of inference of predicate logic. [3] Our system is thus considerably more powerful. Traditionally, modal logic has restricted itself to the propositional calculus since the introduction of quantifiers poses so many problems ([Marcus 1971], [Kaplan 1971], [Hughes and Cresswell 1968]). Of course, not much can be said without predicate calculus; Moore's system, however, lets in predicate calculus through the back door since the actual proofs of theorems are carried out in the first order theory of possible worlds. We, on the other hand, have to explicitly assert all the axioms of predicate calculus and assume that all agents can and do reason with these principles. These axioms are strong enough to prove that all agents know all logical consequences of their knowledge, and that they thus behave like 'perfect reasoners.'

(2) On the other hand, K5 is a good deal weaker than M5. As we have said previously,

3- Of course, the axioms and inference rules are not fixed; we may decide to use a different set, such as is presented in [Kleene 1955], or choose a natural deduction system, as in [Mates 1972], where there are no logical axioms, but more rules of inference. For ease of presentation, we have chosen a system with the minimal number of inference rules.

we do not wish to assume that all agents know all axioms, since there are many axioms on the world that agents simply aren't aware of. We do, however, wish to assume that all agents know all axioms of logic and the basic axioms of knowledge. K5a and K5b capture these constraints, thus avoiding the overly strong assumptions on knowledge that modal logics and possible worlds semantics entail.

As the first step in an integrated theory of knowledge and action, we must provide a solution to Moore's problem: when can we say that an agent knows how to do an action?

Much of Moore's work on this subject is insightful, and it plays a major role in our theory. It is indeed true that an agent knows how to do an action if he knows an executable description for that action. It also seems clear that if an agent knows the general procedure for some parameterized action type and knows in a particular case what the parameters of the action are, that he knows an executable description for that particular action. We contend, however, that it is not in general true that agents know the general procedures for all action types. Instead, we divide the class of actions into *primitive* action types, which are very basic actions such as Put-on and Lift, and *complex* action types such as Bake-Cake and Drive, which are themselves axiomatically defined in terms of primitive actions. We argue that all agents do know the general procedures for the primitive action types. However, because agents in our system do not necessarily know how complex action types are composed out of primitive actions, an agent knows the process for a complex action type only if he knows how that action type is composed out of primitive action types.

We thus say an agent knows how to do an action if

a) the action type is primitive and he knows what the parameters of the action are (knows constants for those parameters) *or* if

b) he knows how the action is built up out of simpler actions and he knows how to perform those simpler actions.

In particular, an agent knows how to do a sequence of actions t1,...,tn if he knows how

to do t1 and completing any action ti in the sequence results in his knowing how to perform the next action. Thus in our system, as in Moore's, an agent need not be able to perform an entire action sequence when he starts to do the action.

It is useful to distinguish between the concepts of *knowing how* to perform an action and *being able* to perform that action. An agent who knows how to perform an action in a particular situation may still be unable to actually perform that action in that situation, because certain other conditions are not satisfied. For example, an agent who knows how to drive a car in S1 may be unable to drive in S1 if no car is available or if the roads are icy. In general, we say that an agent can-perform an action in a situation if he knows how to perform the action, if the physical preconditions for the action are satisfied, and - in the case of an action which involves more than one agent - if certain social protocols governing the behavior of agents are satisfied.

Often, an agent is more interested in being able to achieve some goal than in being able to perform a particular action. For such situations, we introduce the predicate know-how-to-achieve. We say that an agent knows-how-to-achieve a situation with a desired property if he knows of some action that will achieve the desired situation and he can-perform that action.

These three concepts - know-how-to-perform, can-perform, and know-how-to-achieve - are the building blocks of our theory of knowledge and action. Using them, we can solve all of the benchmark problems that Moore's system solved. But these problems - which involved a single agent reasoning about his ability to do an action or sequence of actions, and which demonstrated the knowledge an agent might obtain by performing an action - represent only a small portion of the problems that a robust theory of planning should handle. In a real world situation, agents who do not know how to perform some action do not simply give up, but reason about whether they can obtain the information they need from some other agent, or delegate the action to other more knowledgeable agents. Moore's system could not describe such scenarios since the logic he used was not expressive enough to describe an agent's partial knowledge of another agent's knowledge. Our theory, on the other hand, is

sufficiently expressive for these needs, and we can thus apply ourselves to developing a theory which can handle the aforementioned problems.

4.2. Extensions to Planning

We begin by formally introducing the concept of a plan. Typically, AI planning theories (e.g. [Sacerdoti 1977, McDermott 1985]) have considered only plans constructed out of actions done by a single agent. This concept of planning suffices for simple blocks-world type planning domains, and for restricted classes of problems in simplified multi-agent domains. It does not, however, work well in reasonably complex multi-agent domains in which agents rely upon other agents' actions as they plan. Consider, for example, my plan to get to the airport: Do(I, hail(taxi)) ; Do(driver(taxi),drive(taxi,airport)) ; Do(I, pay(driver(taxi)). The taxi driver's action is an essential part of my plan. We thus define a plan as a structure of events constructed according to specified formation rules, where an event can be any action done by an agent or an 'actorless' event such as a thunderstorm or tornado, and the formation rules correspond to the control structures given in standard real time concurrent programming languages, such as described in [Davis 1984].

Analogous to the concept of can-perform for actions, we introduce the notion of can-execute for plans. An agent a can execute plan p in situation s if he knows in s that he will be able to perform all the actions in the plan for which he is an agent, and can predict that all other events in the plan occur in their proper order. [4] Using this concept, we can now say that an agent can-plan-to-perform an action if he knows of some plan that he can execute which will bring him to a state where he can perform that action. This terminology allows us to describe a situation where, for example, John can plan to call Mary on the phone by asking his friend Bill to tell him her phone number. Likewise, an agent can plan to learn how to make a souffle by registering for classes in a cooking school. Again, an essential part of the agent's plan is his ability to predict the occurrence of the cooking lessons.

4- Note that according to this broad definition, agents can-execute plans that are independently predictable events, such as the sun rising each morning or the president's annual State of the Union address.

Often, an agent who does not know how to do a particular action will be able to delegate the action to another subordinate agent. To formalize this concept, we introduce the predicate 'control', where control(a,b,t) means that agent a controls agent b with respect to a task, or action, t. The control predicate introduces a rather intricate structure on the relationships among agents. To visualize this structure, we can construct a graph by assigning a unique node to each agent, a unique color to each task, and drawing a directed edge of a particular color between two nodes if one agent controls the other with respect to that particular task. Note that while the complete graph is quite complex, the structure that we obtain by considering edges of only one color is simply a forest of trees: a purely hierarchical structure.

We now posit that an agent who controls another agent with respect to a particular task can delegate that task to the controlled agent, and furthermore, that an agent who has been delegated a task will do the task if he possibly can. We can now describe a situation where a supervisor can plan to get an action done by delegating it to his subordinates. Note that he need not know how to do the task that he delegates; he merely must be able to reason that his subordinates know how to do the task. Finally, to integrate the concept of delegation with our theory of planning, we say that an agent can-plan to get an action performed if he either can plan to perform the action himself, or can plan to delegate the action to another agent.

5. Conclusion

We have thus far constructed the foundations for a logic of planning, knowledge, and action. We have demonstrated that our theory solves Moore's benchmark problems (see [Morgenstern 1986] for details) and have begun to extend our theory so that it can deal with more complex planning problems. Although we are still a long way from completing work on a robust theory of knowledge, action, and communication, we believe that our current logic is flexible and expressive enough to provide the basis for such a theory. There are several reasons to believe this to be true:

[1] Agents in our theory have genuinely differing levels of procedural knowledge.

[2] Agents in our theory can reason about other agents' knowledge. This is true even when an agent has only a vague idea of what the more knowledgeable agent knows.

[3] Strings are a natural and important part of our theory. This will prove useful in developing any theory of communication, since so many communicative actions operate primarily on strings. Moreover, since an agent's knowledge about actions is in the form of strings, it should be easy to describe how an agent teaches another agent to do an action.

In sum, our observations suggest that we will be able to extend our current theory to one that is considerably more complex.

Acknowledgements: I'd like to thank Ernie Davis for his ideas, suggestions, and guidance, and for all the weekly discussions that led up to this paper. Thanks also to Phil Cohen, Jeff Finger, Jerry Hobbs, David Israel, Kurt Konolige, Bob Moore, Stan Rosenschein, and Moshe Vardi for helpful discussions and criticisms.

BIBLIOGRAPHY

Appelt, Douglas: *Planning Natural Language Utterances to Satisfy Multiple Goals,* SRI International Technical Note 259, 1982

Asher, Nicholas and Hans Kamp: 'The Knower Paradox and the Logic of Attitudes,' *Proc. Conf. on Theoretical Aspects of Reasoning about Knowledge,* Monterey, 1986 (this volume)

Davis, Ernest: 'A High Level Real-Time Programming Language,' NYU Technical Report, 1984

Fikes, R.E. and Nils Nilsson: 'STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving,' *Artificial Intelligence,* Vol 2, 1971

Gallin, Daniel: *Intensional and Higher Order Modal Logics,* American Elsevier, New York, 1975

Gupta, Anil: 'Truth and Paradox,' *Journal of Philosophical Logic* Vol. 11, No. 1, 1982

Herzberger, Hans: 'Notes on Naive Semantics,' *Journal of Philosophical Logic* Vol. 11, No. 1, 1982

Hintikka, Jaakko: *Knowledge and Belief,* Cornell University Press, Ithica 1962

Hintikka, Jaakko: 'Semantics for Propositional Attitudes,' in Leonard Linsky, ed. *Reference and Modality,* 1971

Hobbs, Jerry and Robert Moore: *Formal Theories of the Commonsense World,* Ablex Publishing Co., Norwood, 1985

Hughes, G.E. and M.J. Cresswell: *An Introduction to Modal Logic,* Methuen, London, 1968

Kaplan, David and Richard Montague: 'A Paradox Regained,' Notre Dame Journal of Formal Logic, vol.1 no.3, pp.79-90, 1960. Also Chapter 9 in **Richmond Thomason, ed:** *Formal Philosophy*

Kaplan, David: 'Quantifying In,' in **Leonard Linsky, ed.** *Reference and Modality*

Kleene, Stephen C.: *Introduction to Metamathematics,* Van Nostrand, New York, 1952

Konolige, Kurt: 'A First Order Formalization of Knowledge and Action for a Multi-agent Planning System' in J.E.Hays and D.Michie, eds. *Machine Intelligence 10,* 1982

Kripke, Saul: *Naming and Necessity,* Harvard University Press, Cambridge, 1972

Kripke, Saul: 'Outline of a Theory of Truth,' Journal of Philosophy, Vol 72, pp. 690-716, 1975.

Kripke, Saul: 'Semantical Analysis of Modal Logic,' *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik,* Vol. 9, 1963

Kripke, Saul: 'Semantical Considerations on Modal Logic,' *Acta Philosophica Fennica,* fasc. 16, pp.83-94, 1963. Also in **Leonard Linsky, ed:** *Reference and Modality*

Linsky, Leonard, ed: *Reference and Modality,* Oxford University Press, London 1971

Marcus, Ruth Barcan: 'Extensionality' in **Leonard Linsky, ed.** *Reference and Modality*

Mates, Benson: *Elementary Logic,* Oxford University Press, 1972

McCarthy, John and Patrick Hayes: 'Some Philosophical Problems from the Standpoint of Artificial Intelligence' in Bernard Meltzer, ed: *Machine Intelligence 4,* 1969

McDermott, Drew: 'A Temporal Logic for Reasoning About Processes and Plans,' *Cognitive Science,* 1982

McDermott, Drew: 'Reasoning About Plans' in Hobbs and Moore, eds. *Formal Theories of the Commonsense World,* 1985

Mendelson, Eliot: *Introduction to Mathematical Logic,* Van Nostrand, Princeton, 1964

Montague, Richard: 'Syntactical Treatments of Modality with Corollaries on Reflexion Principles and Finite Axiomatizability' in *Acta Philosophica Fennica,* fasc. 16, pp. 153-167 1963. Also in **Richmond Thomason, ed:** *Formal Philosophy*

Moore, Robert: *Reasoning About Knowledge and Action,* SRI Technical Note 191, 1980

Morgenstern, Leora: 'Preliminary Studies Toward a Logic of Knowledge, Action, and Communication,' NYU Technical Note, 1986

Perlis, Don: *Language, Computation, and Reality,* unpublished manuscript, 1981

Sacerdoti, Earl: *A Structure for Plans and Behavior,* American Elsevier, New York 1977

Thomason, Richmond, ed: *Formal Philosophy: Selected Papers of Richard Montague,* Yale University Press, New Haven 1974