

AD-A132 473

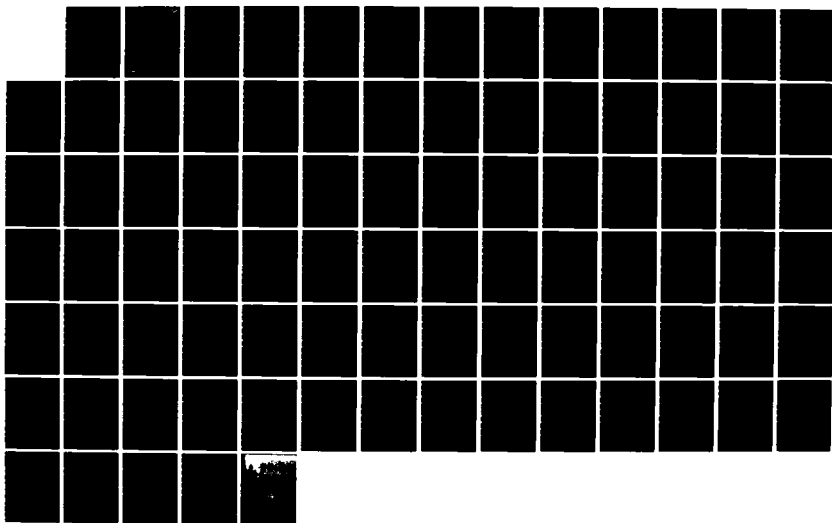
RESEARCH ON DISTRIBUTED ARTIFICIAL INTELLIGENCE(U) SRI
INTERNATIONAL MENLO PARK CA S J ROSENSCHEIN AUG 82
N00014-80-C-0296

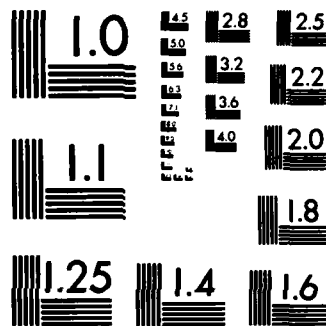
1/1

UNCLASSIFIED

F/G 6/4

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA132473

SRI



International

DTIC FILE COPY

DISTRIBUTION

Approved for publication;
Distribution Unlimited

RESEARCH ON DISTRIBUTED ARTIFICIAL INTELLIGENCE

12

Interim Report

August 1982

By: Stanley J. Rosenschein, Senior Computer Scientist
Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Department of the Navy
Office of Naval Research
Mathematical and Information Sciences Division
800 North Quincy Street
Arlington, Virginia 22217

Attention: Mr. Marvin Denicoff, Code 437
Program Director, Information Systems

Contract No. N00014-80-C-0296
SRI Project 1350

Preparation of this paper was supported by the Office of Naval Research under
Contract N00014-80-C-0296

The views and conclusions contained in this document are those of the authors
and should not be interpreted as representative of the official policies, either
expressed or implied, of the Office of Naval Research or the United States Govern-
ment.

Approved:

Nils J. Nilsson, Director
Artificial Intelligence Center

David H. Brandin, Vice President and Director
Computer Science and Technology Division

DTIC
ELECTE
S SEP 15 1983 **D**

83 09 12 065

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SRI Project 1350	2. GOVT ACCESSION NO. AD-A132473	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Research on Distributed Artificial Intelligence		5. TYPE OF REPORT & PERIOD COVERED Interim 1 Feb 80 - 30 Jun 82
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Stanley J. Rosenschein		8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0296
9. PERFORMING ORGANIZATION NAME AND ADDRESS SRI International Artificial Intelligence Center Menlo Park, CA 94025		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS RR014-08-0A,1-AE NR 610-004
11. CONTROLLING OFFICE NAME AND ADDRESS 		12. REPORT DATE August 1982
		13. NUMBER OF PAGES 81
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research (Code 433) Information Sciences Division 800 N. Quincy St. Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this document is unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Distribution is unlimited.		
18. SUPPLEMENTARY NOTES None		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102- LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BLANK PAGE

Contents

1. Introduction	1
1.1 Planning in a World with other Agents	2
1.2 Reasoning about the Knowledge of other Agents	3
1.3 Planning Communication Acts.	4
1.4 Future Plans	5
2. Plan Synthesis: A Logical Perspective	6
Abstract	6
2.1 Introduction	6
2.2 Preliminaries	8
2.2.1 Syntax	8
2.2.2 Semantics.	8
2.2.3 Axiomatics	9
2.2.4 A Restricted Class of Programs	10
2.3 A Planning Method.	11
2.3.1 Definitions	11
2.3.2 Finding Solutions	12
2.3.2.1 Normal Form for Conditional Plans	12
2.3.2.2 An Algorithm	13
2.3.2.3 Progression and Regression	15
2.3.3 Hierarchical Planning	17
2.4 Discussion.	18
2.4.1 Modeling Actions: The Legacy of STRIPS Operators	18
2.4.2 Nonlinear Planning: Problems with Partial Orders and Shuffles.	19
2.4.3 Hierarchical Planning: Problems with Heuristic Decompositions	20
2.4.4 Some Benefits of Bigression	20
Acknowledgements	21
3. A First Order Formalization of Knowledge and Action for a Multiagent Planning System	22
3.1 Introduction	22
3.1.1 Overview and Related Work.	24
3.2. Agents' Beliefs and First-Order Theories.	26
3.2.1 Metalanguage and Object Language	28
3.2.2 Knowledge and Belief	33
3.2.3 Individuals	34

3.2.4 Knowing Who Someone Is	37
3.2.5 The Object Language as Metalanguage	41
3.3. The Interaction of Actions and Beliefs	45
3.3.1 Situations.	46
3.3.2 Observables.	48
3.3.3 Events Types.	49
3.3.4 Reasoning about Situations and Events.	52
3.3.5 Formalizing Agents' Reasoning about Events.	54
3.3.6 An Example of a Test	56
3.3.7 Plans and Planning	60
3.3.8 Conclusion	63
3.4 Acknowledgments	64
Appendix A: The Wise Man Puzzle.	65
 4. Planning Natural-Language Utterances	 68
4.1 Introduction	68
4.2 Why Plan Utterances?	69
4.3 The KAMP Language Planning System.	70
4.4 Axiomatizing Knowledge about Intensional Concepts	72
4.5 Axiomatizing Linguistic Actions	73
4.6 Conclusion	76
 References	 78

Accession For		
NTIS	GRA&I	☒
DTIC	TAB	☐
Unannounced		☐
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A		



1. Introduction

This document describes research conducted by SRI International on distributed artificial intelligence (DAI) under ONR Contract N00014-80-C-0296 during the period February 1, 1980 to June 30, 1982. In addition to an overview describing the general goals of the DAI project and the major subareas covered, this document includes three papers describing several facets of our research in greater detail.

The ultimate goal of the DAI project is to discover the essential principles upon which construction of a network of cooperative intelligent computer agents could be based. The motivations for distributed AI arise from at least three sources. First, many military problems have an inherently distributed character by virtue of the physical distribution not only of military units themselves but also of the various sensors that provide information affecting those units. Second, the changing economies of computer hardware and communications technology are making distributed systems increasingly cost-effective; since distributed systems with computationally powerful processors will soon be more commonplace, it is worthwhile considering how AI could be used to exploit such systems more fully. Third, there is a strong motivation from within AI. Artificial-intelligence systems that act in an environment must reason about that environment. Thus, when the environment contains other agents, as most realistic environments do, the AI system must reason about the state of those agents and how to affect it. This leads naturally to distributed AI.

The first years of the DAI project at SRI have been devoted to the basic research issues connected with knowledge representation and planning in domains containing multiple agents. Our framework assumes a system containing physical objects and a number of "agents." Each agent is endowed with sets of beliefs, goals, and intentions (plans) operationalized as expressions in suitable language(s). These beliefs, goals, and intentions constitute the "cognitive state" of

the agent, and the overall state of the system consists of the physical state of the objects combined with the cognitive states of the agents. The agents are capable of performing actions (or "operations"), and the next state of the system depends on the actions performed by the several agents.

As with traditional robot problem-solving systems, the choice of operators determines what state transitions can occur. In classical single-agent problem solving, the operators typically affect only the physical state of the system. In contrast thereto, we are primarily interested in operators whose effect is to change the cognitive state of other agents, both as a goal in itself and as a means toward achieving some "physical" goal by enlisting the aid of another agent.

From each agent's point of view, the planning problem consists of forming an intention (plan) over the operators to achieve its goals, given what it believes. The difficulty posed by this problem in the distributed case stems directly from the complexity of cognitive-state-affecting operators. We have been looking at various formalisms for describing cognitive state that would be suitable for practical systems of the type described. In addition, we have been investigating useful sets of operators for achieving interagent cooperation.

1.1. Planning in a World with Other Agents

Most AI planning research to date has been based on a simple state-transition model of action in which there is only one agent and whose actions are always determinate. To handle complex domains realistically, we need to extend the model to allow actions with indeterminate outcomes (especially when outcomes differ in likelihood), as well as actions by more than one agent.

There are several ways of adding indeterminacy to the underlying framework. The situation calculus formulation of planning is able to express indeterminacy up to logical disjunction by simply having the axioms that express the effects of an action contain disjunctive

postconditions. Unfortunately, the STRIPS formulation, which suppresses state variables and expresses the effects of actions as state-description transformations, is incapable of expressing this indeterminacy. Our work on dynamic-logic-based planning addresses this problem by combining some of the best features of STRIPS (e.g. the suppression of state variables and the use of structured search through a space of state descriptions) with the best features of the situation calculus (e.g., the possibility of disjunctive postconditions). Some results of this work are presented in Chapter 2.

The inclusion of concurrency in the formal model can be handled with parameterization of the state transformations by the actions of several agents instead of only one. If we are willing to postulate a global state of the system (even though in practice we may have only incomplete knowledge of this state), we can conceive of the parallel execution of a primitive operation as being a single complex operation on the global state. Reasoning about sequences of actions by the various agents then involves reasoning about interleavings of primitive events. This would be difficult if the only way to perform this reasoning were to enumerate the combinatorially large number of execution sequences and examine each in turn. Fortunately, in typical domains the effects of an action by one agent are ordinarily invariant under most actions by other agents. This fact can be used to facilitate the reasoning. We have investigated extensions of the planning framework that include this form of concurrency.

1.2. Reasoning about the Knowledge of Other Agents

Part of the problem of reasoning about the world involves reasoning about other agents in the world. Part of what we know about other agents has to do with how they act, especially how they act in light of their knowledge and desires. In brief, it is often useful to simplify our description of the behavior of agents by attributing to them certain beliefs and goals which, with the presumption of rationality, predispose them toward certain actions that would tend to further their goals if the world were in accord with their beliefs. Thus, an AI

system that would use the principle of rationality to predict the actions of other agents would need to be able to model those agents' beliefs.

One method for modeling the beliefs of other agents involves using modal logic—i.e., a logic with special operators such as BELIEVE(*p*), where *p* is a sentence, possibly involving the BELIEVE operator as well. The semantics of this operator is often given in terms of possible worlds. That is, what we mean when we say that “*x* believes *p*” is that “*p* is true in all worlds compatible with what *x* believes.” We note that the object of belief is no longer an entity in the logic. For example, we cannot quantify over beliefs. Another possible limitation of the modal approach, at least in its usual form, is that it commits the agent to believing all the consequences of his beliefs. This assumption is often a useful idealization, but is not always appropriate for modeling resource-bounded processes like AI problem-solving systems.

One alternative to the modal approach is often called the “syntactic” approach. In this method the object of belief is taken to be roughly a sentence or a formula of some logic. Many objections have been raised to this approach, chiefly centering around the treatment of quantification. However, a careful formulation of the logic not only avoids most of these problems, but seems to have the advantage of being convenient to implement. Some DAI-related results by Kurt Konolige in this area are reported in Chapter 3.

1.3. Planning Communication Acts

One view of communication is that it involves sending messages that are “dumps” of part of the sender's mental state. In other words, the message “encodes” some of the beliefs of the speaker, and all the hearer has to do is “decode” the message, adding the beliefs to his own belief set. A more sophisticated view holds that messages are carefully built with the intent to affect the state of the hearer in ways far more subtle than simply adding a belief to the hearer's set of beliefs. For instance, in an indirect communication act, the utterance “it is cold in this room” could be recognized as a request to turn up the heat.

To use messages of this complexity, it is necessary for an AI system to contain both a model of the effects of various communication acts and a procedure for synthesizing utterances that achieve desired effects. During the past two years Doug Appelt has investigated this question. The result described in his recent Ph.D. thesis is a system for planning utterances [App82]. A condensed version of these results are presented as Chapter 4.

1.4. Future Plans

During the next year we hope to extend our theoretical understanding of multiagent planning and develop algorithms for consensus formation and plan sharing. We also intend to implement a planning system that is capable of utilizing other similar agents to achieve shared goals. Although we expect the principles underlying this planner to be largely domain-independent we shall select a particular domain for experimentation. We will continue our attempts to characterize a variety of communication-act types using cognitive-state logics, and, to the extent possible, we shall integrate the results into the evolving implementation.

2. Plan Synthesis: A Logical Perspective

This section was written by Stanley Rosenschein.

Abstract

This paper explores some theoretical issues of robot planning from the perspective of propositional dynamic logic. A generalized notion of "progression" and "regression" of conditions through actions is developed. This leads to a bidirectional single-level planning algorithm which is easily extended to hierarchical planning. Multiple pre/post-condition pairs, complex (conjunctive, disjunctive) goals, goals of maintenance and prevention, and plans with tests are all handled in a natural way. The logical framework is used to clarify gaps in existing "non-linear" and "hierarchical" planning strategies.

2.1. Introduction

Although the connection between the A.I. planning problem and automated program synthesis is widely acknowledged, relatively little planning research has made explicit use of concepts from the logic of programs. Such logics, however, offer theoretical insight into various issues in A.I. planning, including compound goals and levels of abstraction [Sac75, Sac77, Tat77, Hay79]. Many of these issues arise in their purest form in domains describable in the propositional calculus (e.g. simple blocks worlds) as evidenced by the literature on the subject [Sus75, Sac75, Wal75]. Thus for clarity and continuity, we choose the propositional setting to develop a unified, abstract treatment of these issues using propositional dynamic logic (PDL) as our primary logical tool [Pra76, Lit77, Pra78a, Pra78b, Har79].

PDL is a decidable modal propositional logic for reasoning about binary state-relations induced by programs. In theory, the existence of such a logic provides an immediate "solution" to the propositional planning problem: One could systematically substitute all possible plans into a schema (the *specification*) which asserts the desired property of the plan. The resulting expressions could be tested for validity to filter out non-solutions. Unfortunately, this fact is of little practical consequence, as such a procedure is certain to be grossly inefficient. The approach developed here imposes additional structure by (1) considering a class of problems that require, in effect, only non-modal reasoning and (2) using suitable "progression" and "regression" operators to structure the search for a solution and allow early pruning of hopeless paths.

Surprisingly, even our restricted formulation covers a more general class of problems than are handled by most comparable A.I. planning methods. For instance, we allow goals to be arbitrary wffs, so disjunctive goals (cited as an unsolved problem by Sacerdoti [Sac77]) require no special treatment at all. The approach provides a theoretical basis for hierarchical plan generation that ties in directly with current ideas on hierarchical program development (see section 1.3.3). In addition, the use of program logic provides a formal basis for specifying and verifying the plan-generating system itself.

Although our work can be generalized along several dimensions (propositional axiom schemata, plans with loops, quantified pre/post-conditions, etc.), these are beyond the scope of the current paper, which focuses instead on the essential structure of the approach. At the same time, it should be noted that the use of axiom schemata seems to be a minimal requirement for a practical application. This paper should be regarded as a foundational study aimed at deepening our understanding of planning; a separate paper will describe implementation considerations [Ros81].

2.2. Preliminaries

This section briefly presents the basic concepts of a loop-free fragment of propositional dynamic logic (PDL). The interested reader is referred to [Pra76, Lit77, Har79] for a more thorough treatment of dynamic logic in its entirety. Though dynamic logic is ordinarily used to reason about *programs*, it is equally appropriate for reasoning about *plans* (in the A.I. sense); thus in this paper the terms *program* and *plan* are used interchangeably.

2.2.1 Syntax

Let P and A denote two symbol sets: *atomic propositions* and *atomic actions*, respectively. Define wffs $P_{P,A}$ and programs $A_{P,A}$ simultaneously (deleting subscripts for convenience):

- (1) $P \subseteq P$
- (2) $A \subseteq A$
- (3) If $p, q \in P$, then $\neg p, p \vee q \in P$
- (4) If $p \in P$ and $\alpha \in A$, then $\langle \alpha \rangle p \in P$
- (5) $\Lambda \in A$
- (6) If $p \in P$ and p is nonmodal (see below), then $p? \in A$
- (7) If $\alpha, \beta \in A$, then $\alpha; \beta \in A$ and $\alpha \cup \beta \in A$

A formula is *nonmodal* if it contains no subformula of the form $\langle \alpha \rangle p$. We abbreviate $\neg(\neg p \vee \neg q)$ as $p \wedge q$, $\neg p \vee q$ as $p \supset q$, $(p \supset q) \wedge (q \supset p)$ as $p \equiv q$, $p \vee \neg p$ as *true*, and $p \wedge \neg p$ as *false*. Parentheses are used when necessary according to the usual conventions.

2.2.2 Semantics

A structure S is a triple (W, π, m) where W is a non-empty set of "worlds," $\pi : P \rightarrow 2^W$, and $m : A \rightarrow 2^{W \times W}$. That is, π assigns to each atomic proposition p the subset of W where

p holds, and m assigns to each atomic action a the binary relation over W representing the next-state relation for a . Given a structure S , meanings can be assigned to arbitrary programs and formulae by extending m and π :

Meanings of Programs

- (1) $m(\Lambda) = \{(s, s) | s \in W\}$ (identity relation over W)
- (2) $m(p?) = \{(s, s) | s \in \pi(p)\}$ (identity relation restricted to worlds where p holds)
- (3) $m(\alpha; \beta) = m(\alpha) \circ m(\beta)$ (composition of relations)
- (4) $m(\alpha \cup \beta) = m(\alpha) \cup m(\beta)$ (union of relations considered as sets)

Meanings of Formulae

- (1) $\pi(\neg p) = W - \pi(p)$
- (2) $\pi(p \vee q) = \pi(p) \cup \pi(q)$
- (3) $\pi(\langle \alpha \rangle p) = \{s \in W | \exists t \in W. (s, t) \in m(\alpha) \wedge t \in \pi(p)\}$

The last equation asserts that $\langle \alpha \rangle p$ is true in those worlds s from which another world t is reachable via α 's next-state relation such that p holds in t . In general our formulas will involve the dual of $\langle \alpha \rangle$, namely $[\alpha]$. $[\alpha]p$ can be read "after α , p ." The intent is that p holds in *all* worlds accessible via α .

A formula p is *valid* in a structure $S = (W, \pi, m)$ (written $S \models p$) iff $\pi(p) = W$; p is *valid* (written $\models p$) iff it is valid in every structure.

2.2.3 Axiomatics

The following system captures the semantics given in the previous section:

Axioms

- (1) Axioms of the propositional calculus
- (2) $[\alpha](p \supset q) \supset ([\alpha]p \supset [\alpha]q)$
- (3) $[\Lambda]p \equiv p$

- (4) $[p?]q \equiv p \supset q$
- (5) $[\alpha; \beta]p \equiv [\alpha][\beta]p$
- (6) $[\alpha \cup \beta]p \equiv [\alpha]p \wedge [\beta]p$

Rules of Inference

- (1) From $p, p \supset q$ derive q (Modus Ponens)
- (2) From p derive $[\alpha]p$ (Necessitation)

If a formula p follows from these axioms under the stated rules of inference, we say it is *provable* and write $\vdash p$; if p can be proved from a set of assumptions, Q , we write $Q \vdash p$.

2.2.4 A Restricted Class of Programs

PDL breaks the ordinary conditional statement into more primitive notions of "test" (?) and "non-deterministic choice" ($\cup$). Though we allow the primitive actions to be non-deterministic, we shall only be interested in deterministic combining forms. Thus we limit the use of ? and $\cup$ to contexts of the form $(p?; \alpha) \cup (\neg p?; \beta)$ and require (for convenience only) that p be atomic. This corresponds to the ordinary conditional, so we abbreviate this program form to $p \rightarrow \alpha, \beta$ and call the class of programs obeying these syntactic restrictions *C-programs* (symbolically $\hat{A}_{P,A}$). The requirement that p be atomic is not restrictive, since arbitrary boolean combinations of tests can be expressed by appropriate use of (possibly nested) conditionals. For example, $\neg p \rightarrow \alpha, \beta$ is equivalent to $p \rightarrow \beta, \alpha$. $(p \wedge q) \rightarrow \alpha, \beta$ is equivalent to $(p \rightarrow (q \rightarrow \alpha, \beta), \beta)$, and so forth.

An important property of our combining forms is that they preserve termination; if the primitives always terminate, every C-program over those primitives will always terminate. (Loops are conspicuously absent.) In PDL, the fact that a program α always terminates is expressed $\langle \alpha \rangle \text{true}$.

2.3. A Planning Method

Having described a suitable language and logic, we are now in a position to discuss planning methods. Section 1.3.1 contains the formal definition of a (single-level) "planning problem" and the corresponding notion of a "solution." This leads directly (section 1.3.2.2) to a bidirectional (single-level) planning algorithm based on "progressing" and "regressing" conditions through actions. Section 1.3.3 describes how the hierarchical planning problem can be regarded as a succession of single-level problems in a way that makes the connection between the levels logically precise.

2.3.1 Definitions

A *planning problem* is a triple $(V, Q, R(u))$ where

$V = (P, A)$ is the *vocabulary* of the problem, consisting of the atomic propositions and the atomic actions.¹

Q is a finite set of axioms, which we will refer to as *domain constraints*. Q is partitioned into two subsets: *static constraints*, which are nonmodal formulae, and *dynamic constraints*, which are always of the form $p \supset [a]q$, p and q being arbitrary nonmodal wffs and a an atomic action. We implicitly assume an axiom of the form $\langle a \rangle \text{true}$ for every atomic action a ; this expresses the fact that the action a always terminates, though Q may only partially specify in what state a terminates. We also assume that Q is consistent.

$R(u)$ is a finite set of formulae called the *plan constraints*. Like the dynamic domain constraints, each of these is of the form $p \supset [u]q$ for nonmodal p and q . The symbol u is a distinguished atomic action not contained in A .

A *solution* to a planning problem $(V, Q, R(u))$ is an expression α in the programming language $\hat{A}_V$ such that for every $r(\alpha)$ (obtained by substituting α for u in R), $Q \vdash r(\alpha)$. I.e., it is provable from the domain constraints Q that α satisfies all the plan constraints.²

¹For some applications it is desirable to constrain the programming language to use only a designated subset of the propositions as tests in conditionals. This requires a straightforward modification of our definition and will not be pursued here.

²Equivalently in semantic terms: Structures that satisfy the domain constraints also satisfy the α -instantiated plan constraints.

(Because of the termination constraints on the atomic actions, α is guaranteed to terminate. So in the language of program logic, we are talking about "total correctness.")

2.3.2 Finding Solutions

Having defined "solutions," we turn our attention to methods for discovering them. A natural way of organizing the search for solutions is to follow the syntactic structure of the programming language.

Recall that a program is either Λ , an atomic action a , or a composite of the form $\alpha; \beta$ or $t \rightarrow \alpha, \beta$ where α and β are programs and t is an atomic proposition. It will simplify the algorithm to consider only programs in a normal form, which we now define.

2.3.2.1 Normal Form for Conditional Plans

A program is in normal form if it consists of a sequence³ of 0 or more atomic actions followed optionally by a conditional program both branches of which are in normal form, followed in turn by 0 or more further atomic actions. More formally, a program is in normal form if it can be written as $A_1; \dots; A_n$, $n \geq 0$, with at most one A_i not atomic, in which case A_i is of the form $t \rightarrow B_1, B_2$ where both B_j are themselves in normal form.⁴ The null sequence is identified with Λ , and we take $\Lambda; \alpha = \alpha; \Lambda = \alpha$.

We have not lost any essential solutions by insisting on this form since every C-program can be put into normal form by transforming the longest (length > 1) sequence of

³Since ";" is associative, we write sequences $a; b; \dots; c$ without indicating order of association.

⁴Warren's method [War76] for introducing conditionals produces plans of an even more restricted form: the conditional must be the *last* action in the sequence. I.e., a plan, once split, may never rejoin. This is not an essential limitation, but it introduces a somewhat greater degree of redundancy than our form. We note in passing that Warren's view of the conditional test as an action, has much in common with PDL's $p?$ action.

steps whose first and last steps are conditionals into a single conditional as follows:

$$(s \rightarrow A, B); \dots; (t \rightarrow C, D) \Rightarrow (s \rightarrow (A; \dots; (t \rightarrow C, D)), (B; \dots; (t \rightarrow C, D)))$$

and applying this transformation recursively to A, B, and the residual $\dots; (t \rightarrow C, D)$.

2.3.2.2 An Algorithm

Suppose we are looking for a normal-form program α that satisfies one of the dynamic constraints $p \supset [\alpha]q$ in R . Consider the following cases corresponding to the possible forms of α :

- (1) $\alpha = \Lambda$. This is a solution if $Q \vdash p \supset q$.
- (2) $\alpha = a; \beta$ or $\alpha = \beta; a$ for some atomic action a . In the former case, α is a solution if $Q \vdash p/a \supset [\beta]q$, where p/a represents the strongest provable post-condition of p and a . Analogously, in the second case, α is a solution if $Q \vdash p \supset [\beta]a \setminus q$ where $a \setminus q$ is the weakest provable precondition of a and q . We call the former case "progression" and the latter "regression."
- (3) $\alpha = t \rightarrow \beta_1, \beta_2$. In this case, α is a solution if $Q \vdash p \wedge t \supset [\beta_1]q$ and $Q \vdash p \wedge \neg t \supset [\beta_2]q$.

We see that (1) defines success, (2) suggests forward and backward strategies for sequential steps, and (3) suggests a forward strategy for conditionals.

In addition, we see that there are several obvious ways to limit the search. First, if $p \supset p/a$, the forward search need not consider action a . (A special case of this arises when $p/a = \text{true}$.) Dually, if $a \setminus q \supset q$, the backward search need not consider a . (Here we have a special case when $a \setminus q = \text{false}$.) These checks eliminate self-loops. We can eliminate cycling in the search space altogether if we are willing to pay the price of checking whether $p_i \supset p/a$ for any p_i in the leading chain of preconditions. Likewise we can check whether $a \setminus q \supset q_j$ for any q_j in the trailing chain of postconditions. Note also that if $p \supset t$ or $p \supset \neg t$, the forward conditional search involving t need not be pursued. p/a can never be *false* since this would

imply failure of a to terminate, contradicting our assumptions about the domain constraints Q .

These observations lead immediately to the following non-deterministic algorithm for computing solutions for the single constraint $p \supset [a]q$: (Multiple constraints will be discussed later.)

Bigression Algorithm

"Bigression" stands for "bidirectional progression and regression." Assume p, q are not false.

$Solve(p, q) = Bigress(p, q, \Lambda, \Lambda)$.

Bigress($pre, post, leader, trailer$):

IF $Q \vdash pre \supset post$ THEN RETURN ($leader; trailer$).

CHOOSE:

CHOOSE $\langle a, pre/a \rangle$ from LiveForward(pre):

RETURN ($Bigress(pre/a, post, leader; a, trailer)$)

CHOOSE $\langle a, a \backslash post \rangle$ from LiveBackward($post$):

RETURN ($Bigress(pre, a \backslash post, leader, a; trailer)$)

CHOOSE t from NonTriv(pre):

RETURN ($leader; C; trailer$)

where $C = (t \rightarrow Bigress(pre \wedge t, post, \Lambda, \Lambda), Bigress(pre \wedge \neg t, post, \Lambda, \Lambda))$

LiveForward(p):

IF $S \neq \emptyset$

where $S = \{ \langle a, p/a \rangle \mid a \in A, Q \not\vdash p \supset p/a \}$

THEN RETURN (S)

ELSE FAIL().

LiveBackward(q):

IF $S \neq \emptyset$

where $S = \{ \langle a, a \backslash q \rangle \mid a \in A, Q \not\vdash a \backslash q \supset q \}$

THEN RETURN (S)

ELSE FAIL().

NonTriv(p):

IF $S \neq \emptyset$

where $S = \{ t \mid t \in P, Q \not\vdash p \supset t, Q \not\vdash p \supset \neg t \}$

THEN RETURN (S)

ELSE FAIL().

The algorithm as presented finds solutions for a single plan constraint. However the extension to the general case is straightforward: To insure that *all* the plan constraints are met, a "Cartesian product" version of this algorithm must be run. A failure in *any* of the constraint components counts as failure and serves to prune that branch.

The bigression algorithm makes use of three additional auxiliary functions: " $Q \vdash$ ", " $/$ ", and " $\backslash$ ". " $Q \vdash$ " is a procedure which takes as input a nonmodal formulae p and decides whether p is provable from Q . If the static axioms, Q_s , are rich enough,⁵ this check can be done using only nonmodal reasoning, i.e., ordinary propositional decision methods. The functions " $/$ " (progression) and " $\backslash$ " (regression) are the subject of the next section.

2.3.2.3 Progression and Regression

Ideally, we would like p/a to compute the strongest postcondition of condition p and action a . Similarly, we would like $a \backslash q$ to compute the weakest precondition [Dij75, Wal75]. In PDL, the weakest precondition of p and a can be expressed simply as $[a]p$, which is obviously the weakest formula implying "after a , p ". The strongest postcondition can be expressed using a "converse" operator which we have not described. (See [Lit77].)

However, given the restricted form of our dynamic axioms, there will be no propositional formula provably equivalent to either of these modal formulae. On the other hand, we can effectively compute the weakest precondition pre and strongest postcondition $post$ for which it is *provable* from Q that pre implies "after a , q " and p implies "after a , $post$." It is these propositional formulae that we label p/a and $a \backslash q$.

⁵Specifically, Q_s must generate all the nonmodal formulae generated by all of Q . In certain pathological cases, such as when Q contains a dynamic axiom of the form $p \supset [a]false$, Q would have to be extended to include extra static axioms, since $p \supset [a]false$ and $\langle a \rangle true$ together imply $\neg p$ — a nonmodal formula derivable only through modal reasoning.

The formula p/a is found by taking the conjunction of the set of formulae each of which is a disjunction of a set of q_i drawn from the "right hand side" of the dynamic axioms of Q ($p_i \supset [a]q_i$) such that the disjunction of the corresponding p_i 's is implied by p . Dually, $a \backslash q$ is found by taking the disjunction of the set of formulae each of which is a conjunction of a set of p_i drawn from the "left hand side" of the dynamic axioms of Q ($p_i \supset [a]q_i$) such that that conjunction of the corresponding q_i 's implies q .

Consider the following sample axioms:

$$\begin{aligned} A &\supset [a](B \vee C) \\ G &\supset [a]B \\ (F \wedge E) &\supset [a]D \end{aligned}$$

In this case, $a \backslash (C \vee D) = (A \wedge G) \vee (F \wedge E)$. The reason for this is that $(B \vee C)$ conjoined with $\neg B$ implies $(C \vee D)$, so the conjunction of the corresponding left-hand sides $(A \wedge G)$ is one disjunct of $a \backslash (C \vee D)$. Likewise, the formula D alone implies $(C \vee D)$, making the corresponding left-hand side $(F \wedge E)$ the second disjunct. These two cases exhaust the possibilities for getting $(C \vee D)$.

The reason why the formulas p/a and $a \backslash q$ defined in this way are not exactly equivalent to strongest postcondition and weakest precondition lies in the nature of our atomic actions. Briefly, in the context of programming languages one typically begins with primitives whose semantics are fully characterized and focuses on characterizing the derived operations (sequencing, etc.) [Dij75] For example, the weakest precondition for the assignment primitive is given by $wp("x := E", P(x)) = P(E)$. This equation asserts that the weakest precondition for condition P and action "x gets E" is precisely P with E substituted for x .

In our case, however, the primitive actions are specified only by axioms giving one-way implications. Thus, unless we make assumptions of a "non-monotonic" nature, we would generally be able to consistently add axioms that "weaken" the precondition or "strengthen" the postcondition of an action. Since "provably weakest" is unattainable, we make do with

"weakest provable." This does not affect the completeness of the search algorithm, since we are only looking for programs which *provably* satisfy the specifications.

2.3.3 Hierarchical Planning

The key observation in extending the single-level algorithm to multi-level, hierarchical planning is that an atomic action at level k is a plan to be solved for at level $k + 1$. This point of view is possible because of the way the planning problem was formalized. Specifically, an atomic action is described by a set of dynamic axioms in Q . Likewise, the desired program is described by a set of dynamic axioms in R . Since the same formal objects, namely sets of dynamic axioms, are involved in both cases, it is natural to assume as primitive some action with given properties at level k and then solve for a program having those properties at level $k + 1$.

Formally, a hierarchical planning problem is a tree of single-level problems. If $\langle V_k = \langle P_k, A_k \rangle, Q_k, R_k(u_k) \rangle$ is the problem at non-leaf node k , then node k has one successor for each $a_{k,i}$ in A_k , and that successor's problem has the form $\langle V_{k+1}, Q_{k+1}, Q'(a_{k,i}) \rangle$ where Q' denotes the subset of dynamic axioms of Q_k having the form $p \supset [a_{k,i}]q$. In other words, the domain constraints on the primitive " a " at level k become plan requirements at level $k + 1$. A solution is a plan using the vocabulary of the leaf nodes that satisfies the requirements of the root node. I.e., α is a solution if it solves $\langle V_n, Q_n, R_1(u_1) \rangle$. The propositional vocabulary and action vocabulary can change from level to level, provided the domain axioms have enough inferential structure to make the transfer from level to level meaningful.

Obviously, for any node k , only the successor nodes corresponding to actions actually used in the solution need be solved. Furthermore, the existence of a solution for each of these nodes guarantees the existence of an overall solution.

As with other hierarchic planners, the main benefit of levels in our approach is heuristic: The choice of intermediate vocabularies and domain axioms constitutes a choice

of "planning islands." Any algorithm that tries to solve a problem by solving the nodes in the hierarchy is, in essence, searching for a plan constrained to go through the states defined by the intermediate actions' domain constraints. The main benefit of logic here is to define a reasonable relation between the levels, namely the relation: "correctly implements."

For a fixed determination of levels and a small number of actions it would be possible to precompute solutions to the subproblems, in which case after solving the problem at the top level, the system would act more like a compiler than a problem solver. In dynamic situations when the lower-level actions (in effect, the "tools" for solving the problem) are changing or when only a small number of actions are ever actually used, it seems more natural to solve subproblems as they arise.

2.4. Discussion

2.4.1 Modeling Actions: The Legacy of STRIPS Operators

Much of the research into the control of planning has been carried out in the STRIPS paradigm. [Fik71, Nil80] In this approach, actions are regarded not as mappings from states to states, but rather as syntactic transformations of state-descriptions to state-descriptions, where state-descriptions are logical formulae. One consequence is the oft-cited benefit of not needing to mention the various "frame conditions," i.e. the properties which are invariant under an action.⁶ Unfortunately, the need for operators to be sensitive to the syntax of state descriptions led researchers to consider only very simple state descriptions (e.g. sets of atomic propositions) and very simple transformations (e.g. add-lists and delete-lists).

⁶However, these invariants need not be as large an obstacle to practical implementation as is commonly supposed (see [Ros81]).

As an example of an action that is difficult to specify with a single add-list/delete-list pair, consider the action *toggle* described by a pair of dynamic axioms:

$$\begin{aligned} On(light) \supset [toggle(switch)] \neg On(light) \\ \neg On(light) \supset [toggle(switch)] On(light) \end{aligned}$$

Since the post-condition depends conditionally on the pre-condition, it cannot be determined in isolation whether *toggle* adds or deletes the wff *On(light)*. The same would hold true for actions with disjunctive post-conditions.

These possibilities notwithstanding, many planning systems do make the assumption that the truth of a given atomic proposition in the state resulting from applying a sequence of operators is a determinate, calculable thing. Techniques which rely crucially on these assumptions are sometimes difficult to adapt to less constraining assumptions. We give two illustrations from NOAH. [Sac75]

2.4.2 Nonlinear Planning: Problems with Partial Orders and Shuffles

The basic idea behind nonlinear planning is the following: To solve a conjunctive goal $G1 \& G2$, find a sequence $S1 = a; b; \dots; c$ which achieves $G1$ and another sequence $S2 = d; e; \dots; f$ which achieves $G2$. Represent the overall plan as a network of partially ordered actions with $S1$ and $S2$ as parallel branches. Now use the "resolve conflicts critic" to detect interference between the plans and impose additional ordering constraints among the actions to rule out the interference. The network encodes the subset of possible shuffles of $S1$ with $S2$ which are believed to achieve the overall goal $G1 \& G2$.

For the resolve conflicts critic to filter interference correctly, it must know what is true at each node of the network. Unfortunately, for nodes that occur after joins, what is true depends crucially on the ultimate linearization of the parallel branches. In the general case, the best that can be done is to represent the disjunction of the strongest postconditions of the

alternative linearizations.⁷ This requires considering the alternatives, of which there are $\binom{m+n}{m}$ where m and n represent the lengths of the action sequences in the two parallel branches. Since it is easy to imagine cases where resolve-conflicts criticism would be an expensive operation, the belief that using a nonlinear strategy is computationally efficient seems to be grounded in the *empirical* hypothesis that operators encountered in practice will permit easy detection of conflicts.

2.4.3 Hierarchical Planning: Problems with Heuristic Decompositions

The justification for partial orderings in NOAH is tied up with a desire not to *prematurely* commit the system to a particular linear order of actions which, though seemingly correct at one level, may expand into incorrect plans at lower levels. This possibility can only arise, of course, if the relation between levels ("plan A achieves the same effect as action a ") is not exact. However, such inexactness undermines the original rationale for hierarchic planning, namely factorization of complexity, since it destroys compositionality and requires that we check complex lower-level plans for "unexpected" *global* interactions. Again, an empirical hypothesis is presumably invoked, namely that by some suitable metric, the plan comes "close" to implementing the abstract action. (It is not immediately obvious, though, what metric could be meaningful for the space in question.)

2.4.4 Some Benefits of Bigression

Some of the benefits of regression were first discussed by Waldinger [Wal75] and appreciated by Warren [War74]. These benefits are reaped dually by including progression, which completes the logical symmetry and allows bidirectional search. As we have described

⁷Actually, NOAH does not represent disjunctive postconditions — which may explain why disjunctive goals are considered problematical.

them, the progression and regression operations handle arbitrary boolean formulas, thus solving conjunctive and disjunctive goals as special cases of a more general strategy. Goals of maintenance and prevention can be incorporated into the algorithm as well by expressing as (nonmodal) wffs the condition to be maintained (m) and the condition to be prevented (v). Since the planning algorithm actually develops a descriptive wff (d) for each state reachable during plan execution, it is straightforward to add a check to the procedures LiveForward, LiveBackward, and NonTriv eliminating paths through states where $Q \vdash d \supset \neg m \vee v$.⁸ This simple approach will work in situations where no dynamic replanning is anticipated; goals of maintenance and prevention involving execution monitoring, feedback and replanning, require more complex strategies.

Acknowledgements

I have profited from discussions with Richard Waldinger, Vaughan Pratt, Kurt Konolige, Dave Wilkins, Jerry Hobbs, and Bob Moore.

⁸For a more thoroughgoing treatment of reasoning about processes with intermediate states, see [Pra78b].

3. A First-Order Formalization of Knowledge and Action for a Multiagent Planning System

This section was written by Kurt Konolige.

3.1. Introduction

We are interested in constructing a computer agent whose behavior will be intelligent enough to perform cooperative tasks involving other agents like itself. The construction of such agents has been a major goal of artificial intelligence research. One of the key tasks such an agent must perform is to form *plans* to carry out its intentions in a complex world in which other planning agents also exist. To construct such agents, it will be necessary to address a number of issues that concern the interaction of knowledge, actions, and planning. Briefly stated, an agent at planning time must take into account what his future states of knowledge will be if he is to form plans that he can execute; and if he must incorporate the plans of other agents into his own, then he must also be able to reason about the knowledge and plans of other agents in an appropriate way. These ideas have been explored by several researchers, especially McCarthy and Hayes [McC89] and Moore [Moo80].

Despite the importance of this problem, there has not been a great deal of work in the area of formalizing a solution. Formalisms for both action and knowledge separately have been examined in some depth, but there have been few attempts at a synthesis. The exception to this is Moore's thesis on reasoning about knowledge and action [Moo80], for which a planner has been recently proposed [App80]. Moore shows how a formalism based on possible-world semantics can be used to reason about the interaction of knowledge and action. In this paper we develop an alternative formalism for reasoning about knowledge, belief, and action; we show how this formalism can be used to deal with several well-known problems, and then describe how it could be used by a plan constructing system.

3.1.1 Overview and Related Work

We seek a formalization of knowing and acting such that a description of their interaction satisfies our intuitions. In the first section, we present a basic formalism for describing an agent's static beliefs about the world. We take a *syntactic* approach here: an agent's beliefs are identified with formulas in a first-order language, called the *object language* (OL). Propositional attitudes such as knowing and wanting are modeled as a relation between an agent and a formula in the OL. By introducing a language (the *metalanguage*, or ML) whose prime object of study is the OL, we are able to describe an agent's beliefs as a set of formulas in the OL, and express partial knowledge of that theory. An agent's reasoning process can be modeled as an inference procedure in the OL: from a base set of facts and rules about the world, he derives a full set of beliefs, called his *theory of the world*.

The syntactic approach to representing propositional attitudes is well-known in the philosophy literature, and in the artificial intelligence field McCarthy [McC79] has developed a closely related approach. The formalism developed here differs mainly in that it explicitly identifies propositional attitudes as relations on sentences in an object language, and uses provability in the OL as the model of an agent's reasoning process. We are able to present quite complex deductions involving the beliefs of agents (see the *Wise Man Puzzle* in Appendix A, for example) by exploiting the technique of *semantic attachment* to model directly an agent's reasoning process. We are indebted to Weyhrauch [Wey80] for an introduction to this technique, and for the general idea of using ML/OL structures to represent agents.

Finally, our work differs from McCarthy's in its careful axiomatization of the relation between ML and OL, and incorporates solutions to several technical problems, including reasoning about *belief-nesting* (beliefs about beliefs; Creary [Cre80] has also described a solution), and a cleaner approach to representing quantified OL expressions in the ML.

(This latter subject is not directly relevant to this paper, and will be reported in [Kon81].)

An alternative to the syntactic approach to representing propositional attitudes is the *possible-world* approach, so-called because it utilizes Kripke-type possible-world semantics for a modal logic of knowledge and belief. Moore [Moo80] has shown how to reason efficiently about propositional attitudes by using a first-order axiomatization of the possible-world semantics for a modal logic. Our objections to the possible-world approach are twofold: first, the possible-world semantics for representing propositional attitudes is complex and at times unintuitive; to deduce facts about an agent's knowledge, one must talk about the possible-worlds that are compatible with what the agent knows. Ultimately, we suspect that the syntactic approach will prove to be a simpler system in which to perform automatic deduction, but further research in both areas is needed to decide this issue. A second objection is that it seems to be difficult to modify possible-world semantics for the modal logic to model adequately inference processes other than logical deduction. The possible-world approach uses the modal axiom that every agent knows the consequences of his knowledge, and this is obviously not true, if only because real agents have resource limitations on their reasoning processes. The syntactic approach does not suffer from this criticism, because it is possible to describe explicitly in the ML the inference procedure an agent might use.

The second part of this paper integrates the syntactic approach to representing knowledge and belief with a *situation calculus* [McC89] description of actions. We concentrate on many of the interactions between knowledge and action presented in Moore's thesis [Moo80]. Simply stated, Moore's account is that an agent's beliefs in any situation arise from at least three sources: direct observation of the world, persistence of beliefs about previous situations, and beliefs about what events led to the current situation. By formalizing this assumption, he shows how to model in an intuitively plausible way the knowledge an agent needs to perform actions, and the knowledge that he gains in performing them. Although

we subscribe to his notions on how knowledge and action should interact, for the reasons stated above we feel that the possible-world approach Moore uses to formalize these ideas, while elegant, may not have the same intuitive appeal as the syntactic approach.

The main contribution of this paper is to show that the syntactic approach, when integrated with a situational calculus description of actions, can adequately formalize Moore's criteria for the interaction of knowledge and belief. An important benchmark is to formalize the idea of a *test*: an agent can perform an action and observe the result to figure out the state of some unobservable property of the world. We conclude the second section with just such an example.

In the final section we consider the application of these results to a planning system, in particular one that would require an agent to take account of other agents' plans in forming his own. We come to the conclusion that such a planning system may not be significantly different from current situation calculus planners in its method of search, but does require considerably more sophistication in the deductions it performs at each node in that search.

3.2. Agents' Beliefs and First-Order Theories

In this section we lay the basic groundwork for our syntactic approach to representing and reasoning about agents' beliefs. We will model an agent's beliefs about the world as a set of statements (or *theory*) in some first-order language with equality. This is not to say that an agent actually represents the world as a set of first-order statements; we are not concerned here with the details of the internal representation of a computer or human agent with respect to its environment. All we seek is a way of modelling the beliefs of an agent in a manner that will make reasonable predictions about the agent's behavior, and still be formally tractable. To this end we assume that we can represent an agent's beliefs about the world as a set of statements in a first-order language, and model the derivation of new beliefs by an agent as an inference process on those statements.

Consider an example from the blocks-world domain; let A_0 be the name of an agent. A_0 will have some set of beliefs about the state of the blocks-world. We represent A_0 's beliefs as a list of well-formed formulas (wffs) in a first-order language with equality. We call this list of wffs A_0 's *theory of the world*. For example, suppose A_0 believes that block B is on block C , and that he is holding block D . Then we would have:

A_0 's Theory of the Blocks-World

$ON(B, C)$

$HOLDING(A_0, D)$

where ON and $HOLDING$ have the appropriate interpretations.

Besides specific facts about the state of the world, A_0 also has some general rules about the way the world is put together. For instance, A_0 may know the rule that if any block x is on any block y , then the top of y is not clear. Using this rule together with specific beliefs about the world, he may be able to deduce that C is not clear. This can be modeled as a process of extending A_0 's initial set of beliefs about the world to include the deduced information:

<u>A_0's Facts and Rules about the World</u>	$\Rightarrow$	<u>A_0's Theory of the World</u>
$ON(B, C)$		$ON(B, C)$
$HOLDING(A_0, D)$		$HOLDING(A_0, D)$
$\forall xy ON(x, y) \supset \sim CLEAR(y)$		$\forall xy ON(x, y) \supset \sim CLEAR(y)$
		$\sim CLEAR(C)$
		...

Thus an agent's theory of the world will be the closure of a set of facts and rules about the world, under some suitably defined inference procedure. We will call the set of basic facts and rules from which all other beliefs are derived the *base set* of the theory. Note that the inference procedure that derives the consequences of the base set need not be logical

deduction; it is readily demonstrated that people do not know all the consequences of their beliefs, that they derive contradictory consequences, etc. We recognize that the problem of deriving the consequences of beliefs for more realistic inference procedures is a thorny and unsolved one, and do not intend to pursue it here. For the purposes of this paper we have chosen logical deduction as the inferential procedure: an agent will be able to deduce the logical consequences of his beliefs.

3.2.1 Metalanguage and Object Language

If we were always to have complete knowledge of an agent's beliefs, then it would be possible to use a simple list of facts and rules to represent the base set of those beliefs. However, it is often the case that our knowledge is incomplete; we may know that an agent either believes fact P or fact Q , but we don't know which. Such a description of an agent's beliefs cannot be modeled by a list of facts. So the modelling process must be extended to a *description* of an agent's beliefs. Since beliefs are wffs in a first-order language, a *metalanguage* can be used to describe a collection of such wffs [Kle67]. The basic idea is to have terms in the metalanguage to denote syntactic expressions in the first-order language used to encode an agent's beliefs. The latter first-order language is called the *object language*, or OL, since it is the object of study of the metalanguage (ML). Predicates in the metalanguage are used to state that an expression of the object language is in an agent's theory of the world. The full expressive power of the metalanguage is available for describing a given theory of the object language.

It is natural to choose a first-order language for the metalanguage, since we will be interested in proof procedures in the ML as well as the OL. Let ML be a sorted, first-order language with variables restricted to range over particular sorts. The domain of discourse of the ML will be both the syntactic expressions of the ML, as well as the domain of discourse of the OL. Thus the ML will be able to state relationships that hold between OL expressions and the actual state of the world.

A basic division of sorts of the ML is between terms that denote individuals in the world, and terms that denote expressions in the OL. Among the former will be terms that denote agents ($A_0, A_1, \dots$) and agents' theories of the world; these will be called T_I terms. We will use the function th of one argument, an agent, to denote that agent's theory of the world.

The other major sort of terms will denote *formulas* of the OL; these will be referred to as T_F terms. Restricting our attention for the moment to sentential formulas of OL, there will be terms in ML that denote propositional letters in OL, and constructors in ML for putting together more complicated formulas from these letters. For example, P' in ML denotes the propositional letter P of the OL,¹ and the ML term $and(P', Q')$ denotes the sentence $P \wedge Q$ of the OL. These ML constructors form an *abstract syntax* [McC62] for OL expressions.

Writing names of formulas using *and*, *or*, *not*, and *imp* as constructors is somewhat cumbersome. For the most part we will use a syntactic abbreviation, enclosing an OL formula in sense quotes,² to indicate that the standard ML term for that formula is intended. For example, we will write:

$$\begin{aligned} \lceil P \wedge Q \rceil & \quad \text{for} \quad and(P', Q') \\ \lceil P \supset (Q \vee R) \rceil & \quad \text{for} \quad imp(P', or(Q', R')) \\ & \quad \text{and so on} \end{aligned}$$

The rule for translating sense-quote abbreviations into T_F terms of the ML is to replace each predicate symbol P of the sense-quote expression by the ML term symbol P' , and each boolean connective by the corresponding ML boolean constructor. As more sorts are introduced into the ML we will extend the sense-quote convention in various ways.

Finally, we introduce the ML predicates *TRUE*, *FACT*, and *PR*, each of which has

¹ The general convention will be to use primed terms in ML to denote the corresponding unprimed formulas in OL.

² They are called sense-quotes to indicate that the sense of the expression is wanted, rather than its truth-value. In [Kap71] these are called *Frege quotes*.

an OL formula as one of its arguments. $TRUE(f)$, where f is an OL formula, means that f is actually true in the world under consideration. It is often the case that we will want to describe a certain condition actually holding in the world, independent of whether some agent believes it or not; for instance, this is critical to our reasoning about events in the next section, where events are defined as transformations from one state of the world to another.

We intend $TRUE$ to have the normal Tarskian definition of truth, so that the truth-recursion axioms are valid. Let the variables f and g range over OL expressions. Then we can write the metalanguage axioms for truth-recursion in the object language as follows:

$$\begin{aligned}
 \forall f \sim TRUE(f) &\equiv TRUE(not(f)) \\
 \forall f g TRUE(f) \vee TRUE(g) &\equiv TRUE(or(f, g)) \\
 \forall f g TRUE(f) \wedge TRUE(g) &\equiv TRUE(and(f, g)) \\
 \forall f g TRUE(f) \supset TRUE(g) &\equiv TRUE(imp(f, g))
 \end{aligned}
 \tag{TR}$$

$FACT(t, f)$, where t is an OL theory, means that f is one of the base set formulas of the theory (and from which the rest of the theory will be derived by deduction). Using $FACT$, agent A_0 's previously exhibited beliefs about the world could be described by the following ML predicates:

$$\begin{aligned}
 &FACT(th(A_0), \lceil ON(B, C) \rceil) \\
 &FACT(th(A_0), \lceil HOLDING(A_0, D) \rceil) \\
 &FACT(th(A_0), \lceil \forall xy ON(x, y) \supset \sim CLEAR(y) \rceil)
 \end{aligned}$$

The last $FACT$ predicate describes a rule that agent A_0 believes.

One special type of $FACT$ that we will make frequent use of is a formula known to all agents. We define the predicate $CFACT$ on OL expressions to mean that a true expression is a $FACT$ for all agents, that is, a *Common FACT*:

$$\forall f CFACT(f) \supset \forall a FACT(th(a), f) \wedge TRUE(f) \quad (CF1)$$

CF doesn't completely axiomatize what we intend a common fact to be, however, since it doesn't say that every agent knows that every agent knows that every agent knows *f*, etc.³ But a fuller characterization of *CFACT* must wait until the technical machinery for describing belief-nesting is developed in a later subsection.

PR(*t*, *f*) means that *f* is provable in the theory *t*. As discussed previously, we will assume that *PR* gives the closure of sentences in OL that can be generated by logical deduction from an original set of *FACT*s. A simple axiomatization of *PR* can be given for Hilbert-style (assumption-free) proofs. There is only one rule of inference, *Modus Ponens*:

$$\forall tfg PR(t, imp(f, g)) \wedge PR(t, f) \supset PR(t, g) \quad (MP)$$

that is, from $P \supset Q$ and *P* in the OL, infer *Q*. Since every *FACT* is an initial theorem of the theory, we assert that each of these is provable:

$$\forall t f FACT(t, f) \supset PR(t, f). \quad (FP)$$

And in each theory the logical axioms of a Hilbert system need to be asserted; we assume a sufficient set for the sentential case.

MP and the Hilbert axioms will be used in ML proofs of the provability of OL statements; these axioms simulate a Hilbert-type proof system for an OL theory. This simulation is necessary because in general there will be an incomplete ML description of the OL theory, rather than a simple list of *FACT*s for that theory. In those special cases when a list of *FACT*s is available, it is possible to run the proof procedure on the OL theory directly. That is, since the intended meaning of the *PR* predicate is provability in the OL theory, we can check whether the *PR* predicate holds in the ML by running a theorem-prover in the OL. It also isn't necessary to use a Hilbert system, and we will

³ Common facts are meant to be the same as the "any fool" concept of [McC78].

feel free to exploit any system of natural deduction that is sound. The technique of using a computable model of the intended interpretation of a predicate to determine the truth of formulas involving that predicate is called *semantic attachment* [Wey80], and it will be used extensively to simplify proofs in later sections.

The provability predicate PR does not have the same characteristics as $TRUE$, and this is important in representing beliefs. For example, the fact that P is not provable doesn't imply that $\sim P$ is provable. If we identify provability with belief, $\sim PR(th(A_0), \lceil P \rceil)$ asserts that P is not one of A_0 's beliefs about the world, but this does not imply $PR(th(A_0), \lceil \sim P \rceil)$, i.e., that A_0 believes $\sim P$. Also, it is possible to express that either A_0 believes that C is clear, or he believes that C is not clear:

$$PR(th(A_0), \lceil CLEAR(C) \rceil) \vee PR(th(A_0), \lceil \sim CLEAR(C) \rceil);$$

this says something quite different from $PR(th(A_0), \lceil CLEAR(C) \vee \sim CLEAR(C) \rceil)$; the latter is a tautology that every agent believes, while the former says something a lot stronger about A_0 's beliefs about the world.

Paralleling the truth recursion axioms TR , we can state rules for the provability of compound OL expressions in terms of their immediate subexpressions. Because of the nature of provability, the axioms for negation, disjunction, and implication, unlike their truth-theoretic counterparts, are not equivalences.

$$\begin{aligned} \forall t f \sim PR(t, f) &\subset PR(t, not(f)) \\ \forall t f g [PR(t, f) \vee PR(t, g)] &\supset PR(t, or(f, g)) \\ \forall t f g [PR(t, f) \wedge PR(t, g)] &\equiv PR(t, and(f, g)) \\ \forall t f g [PR(t, f) \supset PR(t, g)] &\subset PR(t, imp(f, g)) \end{aligned} \tag{PR}$$

These are all deducible from the logical axioms in the Hilbert proof system; for instance, the last assertion is just a restatement of *Modus Ponens*.

Another interesting connection between the PR and $TRUE$ predicates can be drawn by looking at models of the OL. Suppose we have used $FACT$ and PR to describe an

agent's theory T of the world. There will be some set of models that satisfy T , i.e., for which all of T 's theorems hold. The actual world will be one of these models just in case all T 's theorems hold for the world. This condition is statable in the ML as:

$$\forall f PR(T, f) \supset TRUE(f)$$

In general this assertion will not be valid, that is, an agent's beliefs need not correspond to the actual world. By introducing the predicate $TRUE$ in the ML, we are able to state the correspondence between a given theory of the world and the actual state of affairs in the world.

3.2.2 Knowledge and Belief

The PR and $TRUE$ predicates can be used to state our fundamental definitions of knowing and believing for an agent. $BEL(a, f)$ means that agent a believes f ; $KNOW(a, f)$ means that agent a knows f . Then we have the definitions:

$$\begin{aligned} \forall a f BEL(a, f) &\equiv PR(th(a), f) \\ \forall a f KNOW(a, f) &\equiv BEL(a, f) \wedge TRUE(f) \end{aligned} \tag{B1}$$

That is, we identify belief with provability in an OL theory, and knowledge as a belief that actually holds in the world. In model-theoretic terms, a sentence is known to an agent if the sentence holds in all of his models, and the actual world is a model for that sentence. The definition of a common fact in $CF1$ means that all common facts are known to all agents.

We already know that the inference process used in deriving new beliefs from old ones is only approximated as logical consequence, yet we should still expect this approximation to correctly model some of the characteristics we attribute to belief. For instance, if a rational agent believes that $P \supset Q$, and he doesn't believe Q , then it should be the case that he doesn't believe P . Translating to the above notation yields the sentence:

$$BEL(A_0, \lceil P \supset Q \rceil) \wedge \sim BEL(A_0, \lceil Q \rceil) \supset \sim BEL(A_0, \lceil P \rceil)$$

To illustrate the use of axioms for belief and provability given so far, we exhibit a natural deduction proof of this sentence in ML:

- | | | |
|----|---|--------------------|
| 1. | $BEL(A_0, \lceil P \supset Q \rceil)$ | given |
| 2. | $PR(th(A_0), \lceil P \supset Q \rceil)$ | 1,B1 |
| 3. | $\sim BEL(A_0, \lceil Q \rceil)$ | given |
| 4. | $\sim PR(th(A_0), \lceil Q \rceil)$ | 3,B1 |
| 5. | $PR(th(A_0), \lceil P \rceil) \supset PR(th(A_0), \lceil Q \rceil)$ | 2,PR |
| 6. | $\sim PR(th(A_0), \lceil P \rceil)$ | 4,5 contrapositive |
| 7. | $\sim BEL(A_0, \lceil P \rceil)$ | 6,B1 |

This particular proof in the ML cannot be done by semantic attachment to the OL, because it involves reasoning about what isn't provable in the OL theory.

At this point we have presented the basic ideas and definitions for a syntactic approach to representing and reasoning about agents' beliefs. The rest of this section is devoted to exploring various technical issues that arise when extending the previous analysis to talking about individuals.

3.2.3 Individuals

By restricting ourselves to the case of sentential formulas in OL, we have been able to present the basic concepts for representing the beliefs of an agent more simply. Additional complications arise when dealing with terms in the OL that denote individuals rather than truth-values. But a ML encoding of these terms is necessary in order to express such concepts as *agent A_0 knows who B is*.

To talk about the individuals that the OL refers to, we introduce an additional sort into the ML, whose denotation will be the *function terms* of the OL. This sort will be called

T_T , and consists of the following members:

- (1) variables $\alpha, \beta, \dots$;
 - (2) $\{f^n(t_1, \dots, t_n)\}$, where $t_i \in T_T$;
 - (3) $\eta(t)$, where $t \in T_I$ [the "standard name" function];
 - (4) nothing else.
- (TT)

The ML variables $\alpha, \beta, \dots$, range over OL function terms. For example, we can state that A_0 believes a particular block is on C by asserting the ML expression:

$$\exists \alpha \text{ BEL}(A_0, ON'(\alpha, C')).$$

In this expression there are two ML terms in T_T , namely, α and C' . C' is a 0-ary function (or constant) in T_T that denotes the constant term C in OL.⁴ ON' is a type of ML term that hasn't been used explicitly before; it is a member of T_F because it names an OL formula. It takes two arguments, each of which is a ML term denoting an OL term, and constructs an OL formula that is the OL predicate ON of these arguments. So the ML term $ON'(\alpha, C')$ denotes the OL expression $ON(A, C)$, where A is the OL term denoted by α .

It is now possible to give a full definition of T_F terms:

- (1) variables $f, g, \dots$;
 - (2) $\{f^n(t_1, \dots, t_n)\}$, where $t_i \in T_F$ [boolean constructors, e.g., *and*];
 - (3) $\{g^n(t_1, \dots, t_n)\}$, where $t_i \in T_T$ [predicate constructors, e.g., ON'];
 - (4) nothing else.
- (TF)

and T_I terms:

⁴We extend the prime convention to cover ML terms in T_T as well as T_F ; that is, t' in ML denotes the unprimed term t in OL.

- (1) variables $x, y, \dots$;
 - (2) $\{f^n(t_1, \dots, t_n)\}$, where $t_i \in T_I$ [individual constants and functions];
 - (3) $\Delta(t)$, where $t \in T_T$ [the denotation function];
 - (4) nothing else.
- (TI)

We will also find it convenient to extend the notion of sense-quote abbreviations to handle ML terms involving T_T variables. The previous rules are expanded in the following way: all function symbols in the sense-quote expression are replaced by their primed forms, while any symbols used as variables in the surrounding ML expression remain unchanged. For example, the sense-quote expression in $\exists \alpha \text{ KNOW}(A_0, \ulcorner \text{ON}(\alpha, b(C)) \urcorner)$ is to be understood as a syntactic abbreviation for the ML term $\text{ON}'(\alpha, b'(C'))$. We have not yet said what happens to T_I variables in sense-quote expressions; this must wait until standard names are explained in the next subsection.

The introduction of T_T terms into the ML completes the descriptive power of ML for OL expressions. It also lets us handle some of the well-known denotational puzzles in the philosophy literature. One of the simplest of these is the Morningstar-Eveningstar description problem. Both Morningstar and Eveningstar are actually the planet Venus seen at different times of the day. An agent A_0 believes that they are not the same; further, he doesn't have any knowledge about either being the planet Venus. Let MS , ES , and $VENUS$ be OL terms that denote the Morningstar, the Eveningstar, and Venus, respectively. The following set of ML formulas describes this situation:

$$\begin{aligned}
 & \text{TRUE}(\ulcorner ES = VENUS \urcorner) \\
 & \text{TRUE}(\ulcorner MS = VENUS \urcorner) \\
 & \text{BEL}(A_0, \ulcorner MS \neq ES \urcorner) \\
 & \sim \text{BEL}(A_0, \ulcorner ES = VENUS \urcorner) \\
 & \sim \text{BEL}(A_0, \ulcorner MS = VENUS \urcorner)
 \end{aligned}$$

It is perhaps easiest to explain this set of sentences in model-theoretic terms. The intended interpretation of the OL terms ES , MS , and $VENUS$ is the same object, namely

the planet *VENUS*. The two *TRUE* predicates establish this, since they assert that these three terms denote the same individual in the world. On the other hand, the first *BEL* predicate asserts that in the models of A_0 's theory of the world, *MS* and *ES* denote different individuals. This means that the actual world cannot be among the models of this theory. Further, the last two *BEL* predicates assert that *ES* and *MS* are not provably equal to *VENUS* in this theory; hence there will be some models of the theory for which $ES = VENUS$ holds, some for which $MS = VENUS$ holds, and some for which neither holds. From this we conclude that not only is A_0 mistaken as to the equality of *ES* and *MS*, he also is unsure about whether either is the same as *VENUS*. [McC79] lists some other philosophical puzzles that can be handled in a syntactic formulation.

3.2.4 Knowing Who Someone Is

One of the problems that any formal treatment of belief must confront is that of describing when an agent knows who or what something is. For example, the following two English sentences say something very different about the state of A_0 's knowledge:⁵

- (1) " A_0 knows who murdered John."
- (2) " A_0 knows that someone murdered John."

The police would certainly be interested in talking to A_0 if the first statement were true, while the second statement just means that A_0 read the local tabloid. We might paraphrase the first statement by saying that there is some individual who murdered John, and A_0 knows who that individual is. The second statement can be true without A_0 having any knowledge about the particular individual involved in the murder.

How is the distinction between the two sentences above to be realized in this formalism? The second sentence is easy to represent:

$$BEL(A_0, \lceil \exists x \text{ MURDERED}(x, \text{JOHN}) \rceil) \quad (W1)$$

⁵ A similar problem appears in [Qui71]

This simply says that A_0 believes in the existence of an individual who murdered John. It might be supposed that the first sentence could be represented in the following way:

$$\exists \alpha \text{ BEL}(A_0, \lceil \text{MURDERED}(\alpha, \text{JOHN}) \rceil) \quad (W2)$$

$W2$ says that there is a *MURDERED* predicate in A_0 's theory of the world relating some individual (α 's denotation) and John. Unfortunately, this isn't quite strong enough; if the denotation of α is the OL term *murderer*(*JOHN*), then $W2$ is virtually a tautology, and doesn't say that A_0 knows who murdered John. Indeed, if the OL expression in $W1$ is skolemized, it becomes obvious that $W1$ and $W2$ are equivalent.

What seems to be going on here is that different names have a different status as far as identifying individuals is concerned. "Bill" is a sufficient description for identifying John's murderer, whereas "John's murderer" is not. The question of what constitutes a sufficient description is still being debated in the philosophical literature. But for the purposes of this paper, it will suffice if we have a name that is guaranteed to denote the same individual in every model of the OL. By asserting a predicate involving this name in A_0 's theory of the world, it will be possible to encode the fact that A_0 believes that predicate for the given individual. Names that always denote the same individual are called *standard names*.

The formal method of establishing standard names is straightforward. Consider the set of all individuals involved in the situation we wish to consider.⁶ Include in the OL a set of constant symbols, the *standard name symbols*, to be put in one-one correspondence with these individuals. The language OL will be partially interpreted by specifying this correspondence as part of any model of the language; this means that the only models of OL we will consider are those that are faithful to the standard name mapping.

In the metalanguage, we introduce the *standard name function* η of one argument (see the definition of T_T terms above). This function returns the standard name of its argument. Generally we will use lower case Greek letters from the later part of the alphabet as ML

⁶ We restrict ourselves to countable sets here.

variables for OL standard names $[\mu, \nu, \dots]$. The metalanguage statement of " A_0 knows who the murderer of John is" then becomes:

$$\exists x \mu (\eta(x) = \mu) \wedge KNOW(A_0, \lceil MURDERED(\mu, JOHN) \rceil) \quad (W3)$$

Because μ denotes a standard name, the only models of this statement are those in which the same individual x murdered John. This is in contrast to $W1$ and $W2$ above, which allow models in which any individual murdered John. An immediate consequence is that $W1$ and $W2$ are derivable from $W3$, but not the other way around.

So in order to assert that A_0 knows who or what some individual B is, we write in the ML:⁷

$$\exists x \mu (\eta(x) = \mu) \wedge KNOW(A_0, \lceil B = \mu \rceil)$$

By modifying the sense-quote translation rules slightly, it is possible to write OL expressions involving standard names much more compactly. The modification is to assume that any ML variable of type T_I occurring within a sense-quote gets translated to the standard name of that variable. With this rule, for example, the above assertion comes out as $\exists x KNOW(A_0, \lceil B = x \rceil)$.

We will use the predicate $KNOWIS(a, \beta)$ to mean that the agent a knows who or what the OL term denoted by β refers to. The definition of $KNOWIS$ is:

$$\forall a \beta KNOWIS(a, \beta) \equiv \exists x KNOW(a, \lceil \beta = x \rceil) \quad (KW)$$

Note that the property of being a standard name is a relation between a term of the OL and models of this language, and hence cannot be stated in the OL. The use of a metalanguage allows us to talk about the relation between the OL and its models.

⁷ This analysis essentially follows that of [Kap71], with the extension of standard names to all individuals in the domain, rather than just numbers and a few other abstract objects. There are problems in using standard names for complex individuals, however; see [Kap71].

One of the proof-theoretic consequences of using standard names is that every theory can be augmented with inequalities stating the uniqueness of individuals named by standard names. In the metalanguage, we write:

$$\forall xy \ x \neq y \supset \forall t \ PR(t, \ulcorner x \neq y \urcorner) \quad (SN)$$

Formally, the definition of a standard name can be axiomatized in the ML by introducing the denotation function Δ .⁸ $\Delta(\alpha)$, where α denotes an OL term, is the denotation of α in the actual world; it is the inverse of the standard name function, since it maps an OL term into its denotation. There is an intimate relation between the denotation function and equality statements in OL formulas describing the world:

$$\forall \alpha \beta \ TRUE(\ulcorner \alpha = \beta \urcorner) \equiv \Delta(\alpha) = \Delta(\beta) \quad (D1)$$

that is, two OL terms are equal in the actual world just in case they denote the same individual; D1 can be viewed as a definition of the intended interpretation of equality. The prime purpose of the denotation function is to tie together the denotation of terms in the OL and the ML. For standard names, it can be used to state that the denotation of a standard name is the same individual in all situations, something that cannot be done with equality predicates in the OL:

$$\forall x \ \Delta(\eta(x)) = x \quad (D2)$$

For example, by asserting $\eta(VENUS) = VENUS'$ in ML, we fix the denotation of the OL term $VENUS'$ to be the individual denoted by the ML term $VENUS$ in all models of the OL.

The introduction of standard names with fixed denotations across all models makes

⁸This is Church's denotation predicate in function form [Chu51]; since a term can have only one denotation, it is simpler to use a function.

the task of relating the OL to the ML easier. By introducing this "common coin" for naming individuals, we are able to write expressions of the OL that represent beliefs without constantly worrying about the subtle consequences of the denotational variance of terms in those expressions. Standard names will play an important role in describing belief-nesting (beliefs about beliefs), in describing executable actions, and in simplifying the deduction process.

3.2.5 The Object Language as Metalanguage

In this subsection we extend the OL to include a description of another object language OL' . Thus extended, the OL can be viewed as a metalanguage for OL' . The reason we want to do this is that it will be necessary for representing an agent's view of a world that is changing under the influence of events. In the next section we will show how an agent can model the way in which the world changes by describing what is true about different states of the world connected by events. But to describe these states of the world, or *situations*, the agent's theory must talk about sentences of another language holding in a given situation.

Before trying to extend the formal apparatus of the OL to describe another OL, it is helpful to examine more closely the relation between the ML as a means of studying the OL and as a means of describing the actual world. This is because the structure of a ML/OL pair will be very similar no matter what the depth of embedding; and the simplest such structure to study is obviously the topmost one. Although we initially characterized the ML's domain of discourse as including that of the OL, it appears that we have not made much use of this characterization. In describing the models of OL, however, it was necessary to pick out the model that was the actual world; this was done with the predicate *TRUE*. And it was impossible to state the definition of a standard name without appealing to terms in the ML that referred to individuals in the actual world. So, in fact, we have already used the ML to characterize the actual state of the world and the individuals that

populate it.

We have stated that agents' beliefs are represented as first-order theories of the world. The ML is, by the above argument, just such a theory; but whose theory of the world is it? One useful interpretation is to take what we will call the *egocentric view*: a theory in the ML is identified as the theory of a particular agent. That is, suppose we were to build a computer agent and invest him with a ML/OL structure as a way of representing other agents' beliefs. Then the nonlogical axioms of the ML would constitute the computer agent's theory of the world. The interpretation of the ML predicate *TRUE* would be "what the computer agent believes about the world," and of the predicate *KNOW*, "what another agent believes that agrees with what the computer agent believes." In this interpretation, there is no sense of absolute truth or knowledge; the beliefs of one agent are always judged relative to those of another.

Suppose we identify the agent A_0 with the ML; what interpretation does the OL theory $th(A_0)$ have? Interestingly enough, it is A_0 's introspective description of his own beliefs. Unlike other agents' theories of the world, $th(A_0)$ shares an intimate connection with formulas that hold in the ML. For a rational agent, it should be the case that if he believes P , then he believes that he believes P . We can state this connection by the following rule of inference:

Belief attachment: If the agent a is identified with the ML, then from
 $TRUE(f)$ infer $BEL(th(a), f)$.

Introspection will be useful when we consider planning, because a planning agent must be able to reflect on the future state of his beliefs when carrying out some plan.

If the metalanguage is intended to describe the actual world, then it is reasonable to ask what the relation is between models of the ML and models of its OL, and whether this connection can be formalized in the ML. We start by adding predicate symbols to the ML whose intended meaning is a property of the actual world, rather than of the OL and

its models. Consider such a predicate P of no arguments, and let its intended meaning be "222 Baker Street Apt 13 is unoccupied:" that is, the actual world satisfies P just in case this apartment is indeed unoccupied. In the OL there is also a predicate symbol P of no arguments whose meaning we wish to coincide with that of the ML predicate P . The fact that these symbols are the same is an orthographic accident; they come from different languages and there is thus no inherent connection between them. However, because the ML can describe the syntax and semantics of the OL, it is possible to axiomatize the desired connection. Let P' be the ML term (in T_F) denoting the OL predicate P . Then P in the ML and OL have the same meaning if

$$P \equiv \text{TRUE}(P') \quad (R1)$$

is asserted in the ML. For suppose the actual world satisfies P in the ML; then $\text{TRUE}(P')$ must also hold, and hence by the meaning of TRUE , the actual world is also a model for P in the OL. Similarly, if the actual world falsifies P in the ML, $\text{TRUE}(\text{not}(P'))$ must hold, and the actual world falsifies P in the OL also. So the proposition named by P' holds just in case Apt. 13 at 222 Baker Street is unoccupied, and thus the meanings of P in the ML and P in the OL coincide.

For predicates that have arguments, the connection is complicated by the need to make sure that the terms used in the ML and OL actually refer to the same individuals. So, for example, if P is a ML predicate of two arguments that we wish to mean the same as the OL predicate P , we would write:

$$\forall \alpha \beta \text{ TRUE}(\ulcorner P(\alpha, \beta) \urcorner) \equiv P(\Delta(\alpha), \Delta(\beta)); \quad (R2)$$

that is, since the denotation function Δ gives the individuals denoted by the OL terms α and β , P in the ML agrees with P in the OL on these individuals. Using standard names, $R2$ could be rewritten as

$$\forall xy P(x,y) \equiv TRUE(\ulcorner P(x,y) \urcorner) \quad (R3)$$

since, by *D2*, $\Delta(\eta(x)) = x$, $\Delta(\eta(y)) = y$. Note that the standard name convention for sense-quotes is in force for *R3*.

Using *TRUE* and equivalence, axioms like *R3* cause predicate symbols to have a "standard meaning" across the ML and OL, in much the same way that *D2* formalizes standard names using the denotation function and equality. But while nonstandard names are a useful device for encoding an agent's beliefs about individuals that the agent may have misidentified (recall the Morningstar–Eveningstar example), nonstandard predicates don't seem to serve any useful purpose. So we will assume that for every predicate symbol *P* in ML, there is a function symbol of the form *P'* whose denotation is the OL predicate *P*, and there is an axiom of the form *R2* equating the meaning of these predicates.

To make the OL into a metalanguage for OL', we simply introduce sorts that denote OL' expressions into the OL, in exactly the same way that it was done for the ML. In addition, the various axioms that tie the ML and OL together (*MP*, *D1*, etc.) must also be asserted in the OL. Unfortunately, this also means that the ML itself must have a new set of terms denoting terms in the new OL sorts; the machinery for describing embedded ML/OL chains rapidly becomes confusing as the depth of the embedding grows. So in this paper we will supply just enough of the logical machinery to work through the examples by introducing two conventions; readers who want more detail are referred to [Kon81].

The first convention is an extension of the sense-quote abbreviation to include ML variables of the sort T_F (denoting formulas of the OL). When these occur in sense-quotes, they are to be translated as the *standard name* of the variable; hence they denote the name of an expression. To take an example, we will complete the axiomatization of *CFACT*:

$$\forall f CFACT(f) \supset \forall a KNOW(a, \ulcorner CFACT(f) \urcorner) \quad (CF2)$$

CF2 asserts that if f is a common fact, then every agent knows it is a common fact. The sense-quote term $\lceil CFACT(f) \rceil$ denotes the OL expression $CFACT(f')$, where f' is the standard name of the OL' expression corresponding to f .

The second convention is to allow embedded sense-quotes to form the standard name of an expression, as in

$$\forall f CFACT(\lceil CFACT(f) \rceil \supset CFACT(\lceil CFACT(f) \rceil)). \quad (CF3)$$

Here the embedded sense quotes translate to the standard name for the OL' expression $CFACT(f')$. *CF3* says that common facts will be inherited down to the next level of embedding in the ML/OL chain.

In practice, we hope that the depth of embedding needed to solve a given problem will be small, since the complexity needed for even the three-level structure of ML, OL, and OL' is substantial. Also, the technique of semantic attachment can be used to reduce the complexity of reasoning about embedded structures by attaching to a particular level of an embedded structure and reasoning in that language. In Appendix A we use embedded ML/OL structures to solve the wise man puzzle, which involves reasoning to a depth of embedding of three (ML, OL, and OL'); we exploit semantic attachment to simplify the reasoning involved.

3.3. The Interaction of Actions and Beliefs

The previous section laid the groundwork for a syntactic treatment of knowledge and belief in a static world. This must be integrated with a formal treatment of actions in order to accomplish our original task of formalizing the interaction of knowledge and action. We examine the following two questions:

- What knowledge is required by an agent to successfully perform an action?
- What knowledge does an agent gain in performing an action?

The methodology we will use is to apply the *situation calculus* [McC69] approach to first formally describe the way in which the world changes as events occur. It will then be assumed that this formal system is a reasonable approximation to the way an agent reasons about changes in the world: this means that it becomes part of an agent's rules about the world. By simply attributing a facility for reasoning about events to agents, it turns out that we are able to answer both these questions formally, and that this formalization corresponds well with our intuitions about real agents. This is essentially the same method that was used by Moore in [Moo80]; here, we show that it can be successfully carried out for a syntactic formalization of knowledge and belief.

Once the formal requirements for reasoning about events have been specified, we consider how an agent might plan to achieve a goal using his knowledge of actions. We conclude that planning is inherently a process of self-reflection: that is, in order to construct a plan, an agent must reflect on what the state of his beliefs will be as the plan is undergoing execution. Such a self-reflection process is represented naturally by a ML/OL structure in which the planning agent is identified with the ML, and his future states are theories of the OL. We will show how it is possible to construct plans within this representation, and extend it to include plans that involve other cooperative agents.

3.3.1 Situations

In the situation calculus approach, events are taken to be relations on *situations*, where situations are snapshots of the world at a particular moment in time. It is natural to identify situations with models of a language used to describe the world; in this case, we will use the language OL of the previous section, because the ML for describing models of the OL is already laid out. In the ML, situations will be named by terms, generally the constants $\{S_0, S_1, \dots\}$. A formula f of the OL holds in a situation s when the situation satisfies f ; the ML predicate $H(s, f)$ will be used to indicate this condition. If the situation S_0 is singled out as being the actual world (and the initial world for planning problems),

then *TRUE* can be defined in terms of *H*:

$$\forall f \text{ TRUE}(f) \equiv H(S_0, f). \quad (H1)$$

Since *H* describes satisfiability in a model, the truth-recursion axioms *TR* are valid for *H* as well as *TRUE*.

If we consider agents to be part of the domain of discourse, then their beliefs can change from one situation to the next, just as any other inessential property of an agent might. But if an agent's beliefs change from situation to situation, then the theory that is used to model these beliefs must also change. One way to represent an agent's changing beliefs is to ascribe a different theory to an agent in each situation to model his beliefs in that situation. In the ML, we will write *ths(a, s)* to denote agent *a*'s beliefs in situation *s*; if *S*₀ is taken to be the actual world, then it is obvious that $\forall a \text{ ths}(a, S_0) = th(a)$.

But we might now ask what situation the expressions in each of these theories are about. Suppose that the OL sentence *P* is a member of *ths*(*A*₀, *S*₁), and thus one of *A*₀'s beliefs in situation *S*₁. We would naturally want *P* to be a property that *A*₀ believes to hold of situation *S*₁ (and not *S*₀ or some other situation). That is, *ths(a, s)* represents agent *a*'s beliefs in situation *s*, about situation *s*. In informal usage we will call the situation we are focusing on the *current situation*, and say "the agent *a* in situation *s*" when we are referring to the agent's beliefs in that situation. Later we will show how to represent an agent's beliefs about situations other than the one he is currently in.

For each situation, an agent's beliefs in that situation are specified by a theory. Given this arrangement, we define the new predicates *B* and *K* as similar to *BEL* and *KNOW*, but with a situation argument:

$$\begin{aligned} \forall a s f \text{ B}(a, s, f) &\equiv \text{PR}(\text{ths}(a, s), f) \\ \forall a s f \text{ K}(a, s, f) &\equiv \text{B}(a, s, f) \wedge H(s, f) \end{aligned} \quad (B2)$$

B(a, s, f) means that in situation *s* agent *a* believes that *f* holds in *s*; *K* is similar, with the condition that *f* actually holds in *s*. Note that the underlying predicates *FACT* and *PR*

do not have to be changed, since they are defined on theories of OL rather than models. Thus the properties of *BEL* and *KNOW* described in the previous section also hold for *B* and *K* in any particular situation. *BEL* and *KNOW* can be defined as *B* and *K* in the situation S_0 .

Several extensions to the formalism presented in the first section must be made to deal with situations. A new denotation function δ takes a situation argument as well as an OL term: $\delta(s, \alpha)$ is the denotation of α in situation s . $\Delta(\alpha)$ gives the denotation of a α in situation S_0 , and is definable as $\delta(S_0, \alpha)$. The appropriate forms of *D1* and *D2* are:

$$\begin{aligned} \forall s \alpha \beta H(\ulcorner \alpha = \beta \urcorner) &\equiv [\delta(s, \alpha) = \delta(s, \beta)] \\ \forall s x \delta(s, \eta(x)) &= x \end{aligned} \tag{D3}$$

This last says that standard names always have the same interpretation in every situation. Non-standard names can change their denotation in different situations, e.g., the block denoted by "the block A_0 is holding" may be changed by A_0 's actions.

Finally, we require the appropriate versions of *R1*–*R6*, where these axioms are appropriately generalized to refer to all situations.

3.3.2 Observables

Following Moore [Moo80], we recognize three ways that an agent can acquire beliefs in a situation:

- He can observe the world around him.
- His beliefs about past situations persist in the current situation.
- He can reason about the way in which the current situation arose from events that occurred in previous situations.

In the next few subsections we describe how an agent's beliefs persist and how he reasons about events; here we formalize what it means for a property of the world to be observable.

It is certainly true that there are many properties of the world we live in that are not

directly observable; for example, consider a gas oven whose pilot light is completely encased and hence not visible. Whether this pilot light is on or off isn't an observable property, but there are other observations that could be made to test what the state of the pilot light is, e.g., by turning on the oven and observing whether it lights. What we actually consider to be observable depends on how we formalize a given problem domain; but it is important for a planning agent to be able to make the distinction between properties of the world he can observe directly, and those he must infer.

One of the reasons that it is handy to have a separate theory representing the beliefs of an agent in each situation is that we then have a way of describing the effect of observable properties on an agent's beliefs. Formally, we can state that a property is observable by asserting that in every situation, subject to certain preconditions that are required for the felicitous observation of the property, an agent knows whether that property holds or not. For example, in the OL let o be an oven, and let $LIT(o)$ mean that o is lit. Then $LIT(o)$ is asserted to be observable by:

$$\forall a, o, s \ H(s, \lceil AT(a, o) \rceil) \supset [K(a, s, \lceil LIT(o) \rceil) \vee K(a, s, \lceil \sim LIT(o) \rceil)] \quad (O1)$$

that is, if the agent is actually at the oven, he knows either that it is lit, or that it is not lit. Recall from the previous section on knowledge and belief that this says something very strong about the state of a 's knowledge, and is not derivable from the tautology $K(a, s, \lceil LIT(o) \vee \sim LIT(o) \rceil)$.

3.3.3 Events Types

Event types are relations on situations; a given event type describes the possible states of the world that could result from an event occurring in any initial state. We will use the three-place predicate EV in the metalanguage to describe event types: $EV(e, s_i, s_f)$, where e is an event type and s_i and s_f are situations, means that s_f results from an event of type

e occurring in s_i . An event is an instance of an event type,⁹ but generally we will not have to distinguish them for the purposes of this paper, and we will use "event" for "event type" freely.

Generally the events of interest will be agents' actions, and these will be constructed in the ML using terms representing actions, agents, and the objects involved in the action (the *parameters* of the action). If act is an action, then $do(a, act)$ is the event of agent a performing this action. Consider the situation calculus axiomatization of a simple blocks-world action, $puton(x, y)$, where the parameters of the action are blocks:

$$\begin{aligned} \forall x y s_i s_f \ EV(do(a, puton(x, y)), s_i, s_f) \supset & H(s_i, \lceil CLEAR(y) \rceil) \wedge \\ & H(s_i, \lceil HOLDING(a, x) \rceil) \wedge \\ & H(s_f, \lceil ON(x, y) \rceil) \wedge \\ & H(s_f, \lceil \sim HOLDING(a, x) \rceil) \wedge \end{aligned} \quad (PO1)$$

$$\begin{aligned} \forall x y s_i s_f \ EV(do(a, puton(x, y)), s_i, s_f) \supset \\ [\forall f \ SAF(f) \wedge f \neq \lceil CLEAR(y) \rceil \wedge f \neq \lceil HOLDING(a, x) \rceil \supset \quad (PO2) \\ H(s_i, f) \equiv H(s_f, f)] \end{aligned}$$

The form of *PO1* is an implication, so the right-hand side describes the conditions under which situations s_i and s_f are related by the event of a putting x on y . The first two conjuncts on the right-hand side are essentially preconditions for the event to occur, since they state conditions on the initial situation s_i that must be satisfied for *EV* to hold. The preconditions are that $CLEAR(\eta(y))$ and $HOLDING(\eta(a), \eta(x))$ must hold in situation s_i ; note that the standard names for the parameters are indicated by the sense-quote convention. If the preconditions are not met, then there is no situation s_f that is the successor to s_i under the event e . The rest of the conjuncts describe which formulas of the OL are to hold in the new situation s_f .

⁹For example, "Borg's winning of Wimbledon yesterday was fortuitous" is a statement about a single event, but "Borg winning Wimbledon has happened five times" describes an event type that had five particular instances.

PO2 specifies that all formulas of a certain type that hold in s_i are also to hold in s_f . It is thus a *frame axiom* for the event e , describing which aspects of the situation s_i remain unchanged after the event occurs. The predicate *SAF* stands for Simple Atomic Formula; it picks out those formulas of the OL that are composed of atomic predicates over standard names. Although *SAF* applies only to nonnegated atomic formulas, the frame axiom carries over negated atomic formulas as well, since $H(s, \text{not}(f))$ is equivalent to $\sim H(s, f)$.¹⁰ Among the nicer features of this axiomatization is that events whose outcomes are conditional on the initial state can be easily described. For instance, consider the event of an agent turning on a gas oven that has a pilot light. If the pilot light is on, the oven will be lit; if the pilot light is off, the oven will have whatever status, lit or unlit, it had before the event occurred (the oven may already have been on). Let $PL(o)$ be an OL predicate meaning "the pilot light of oven o is on"; and let $LIT(o)$ mean "oven o is lit." Then the event of an agent turning on o can be described as:

$$\begin{aligned} \forall s_i s_f o \text{ EV}(\text{do}(a, \text{light}(o)), s_i, s_f) \supset \\ H(s_i, \lceil AT(a, o) \rceil) \wedge \\ H(s_i, \lceil PL(o) \rceil) \supset H(s_f, \lceil LIT(o) \rceil) \wedge \\ H(s_i, \lceil \sim PL(o) \rceil) \supset [H(s_f, \lceil LIT(o) \rceil) \equiv H(s_i, \lceil LIT(o) \rceil)] \end{aligned} \quad (LT1)$$

$$\begin{aligned} \forall s_i s_f o \text{ EV}(\text{do}(a, \text{light}(o)), s_i, s_f) \supset \\ [\forall f \text{ SAF}(f) \wedge f \neq \lceil LIT(o) \rceil \supset H(s_i, f) \equiv H(s_f, f)] \end{aligned} \quad (LT2)$$

The second conjunct of *LT1* gives the result of the event in case the pilot light is on: the oven will be lit. The third conjunct says that if the pilot light is off, the oven will be lit in s_f just in case it was lit in s_i , i.e., its status doesn't change. *LT2* is the frame axiom.

3.3.4 Reasoning about Situations and Events

The axiomatization of events as relations on situations enables us to talk about what is

¹⁰ The axiomatization of events given here is a standard one in the AI literature on formal planning, and there are well-known problems involving the use of frame axioms like the one above. We are not attempting to add any new insight to this particular aspect of planning; but we are interested in having a formal description of events to integrate with our theory of belief, and this seems to be the best formulation currently available.

true in the world after some events have occurred starting from an initial situation (which we will generally take to be S_0). What it doesn't tell us is how an agent's beliefs about the world will change; nothing in the *PO* or *LT* axioms gives any insight into this. It might be suspected that, as events are described by axioms as changing the actual state of the world, this description might be extended to cover agents' theories as well, e.g., changing A_0 's theory in situation S_0 ($ths(A_0, S_0)$) into his theory in situation S_1 ($ths(A_0, S_1)$).¹¹ But there is no obvious or well-motivated way to make modifications to axioms like *PO* and *LT* so that they take into account agents' beliefs about a situation rather than what actually holds in the situation.¹² What is needed here is a principled way of deriving the changes to an agent's beliefs that result from an event, given a description of the event as a relation on situations. Credit for the recognition of this problem belongs to Robert Moore, and we will formalize the solution he presented in his thesis, the main points of which follow.

The solution to this difficulty lies in making the observation that agents are reasoning entities. Consider how agent A_0 might reason about some event E ; let us suppose the event is that agent A_0 turned on the oven in situation S_0 , and that the result was that the oven was not lit in situation S_1 . What should A_0 's beliefs be in situation S_1 ? First, by observation, he knows that the oven isn't lit. He also believes (in S_1) that the current situation resulted from the event E occurring in situation S_0 . So A_0 reasons as follows: if, in situation S_0 , the pilot light of the oven had been on, then in S_1 the oven would be lit, since he turned it on. But the oven isn't lit; hence the pilot light couldn't have been on in S_0 , and remains not on in S_1 .

There are several important things to note about this analysis. The first is that, as

¹¹ Indeed, it might be thought that the most widely known AI planning system, *STRIPS*, has just such a mechanism in its add/delete list approach to describing events. However, closer examination reveals that because *STRIPS* makes the assumption that it has a *partial model* in the sense of [Wey80], it is actually slightly less descriptive than the situational approach described above [Nil80].

¹² There is one proposal that is suggested by our use of H to refer to the actual situation and PR to statements that an agent believes about a situation, namely, to replace all predicates involving H with the corresponding ones involving PR . However, it can be shown that the substitution of $PR(th(A_0, s), \dots)$ for $H(s, \dots)$ yields counterintuitive results for A_0 's beliefs.

suggested previously, A_0 's beliefs in situation S_1 comes from only three sources: observations ("the oven is not lit"), persistence of beliefs about previous situations ("if in S_0 the pilot light had been on..."), and beliefs about the way events change the world. This latter is equivalent to having some form of *PO1* as part of A_0 's beliefs in situation S_1 . From these three sources A_0 is able to generate a new set of beliefs for S_1 .

The second thing to note is that none of A_0 's reasoning in S_1 could have taken place unless he believed that S_1 resulted from S_0 via the event E . Beliefs about what sequence of events led to the current situation play a very important role in reasoning about that situation, and, like other beliefs, they can be mistaken or inferred from other evidence. Suppose, for example, that A_0 suddenly sees the oven become lit. He might infer that the only way that could happen when it wasn't previously lit would be for an agent to turn it on; this is inferring that the situation where the oven is lit is connected by a certain event with a previous situation where the oven wasn't lit. We will not be concerned with this kind of inference here, although we note the possibility of doing *event recognition* in this framework. The events we are interested in are actions, and the assumption we will make for the remainder of this paper is that an agent knows what action it is that he performs in executing a plan.

A third aspect of this reasoning that is unusual is that the axiomatization of events is being used in a different way than a planning program would normally consider doing. Typically, a planner uses an event description like *LT1* to form plans to light the oven, and the side condition that the pilot light be on is one of the things that can go wrong with the plan, and so must be taken into account as a subgoal. However, in the above example A_0 has used *LT1* to reason about a property of the world that is not available to his direct observation, that is, as a test. This is an important characteristic for any formalism that combines a description of agents' beliefs with a description of events; a single description of an event should suffice for an agent to reason about it either as a means of effecting a change in the world, or as a test that adds to his beliefs about the world.

Finally, the precondition that A_0 be at the oven to turn it on translates naturally in this analysis into a precondition on A_0 's beliefs in situation S_0 . If A_0 is to reason that situation S_1 is the successor to S_0 under the event E , he must believe that he was actually at the oven in situation S_0 . For if he doesn't believe this, then he cannot use *PO1* to infer anything about the results of his action.

We might summarize the analysis of this section in the following way: by making the simple assumption that an agent reasons about the way in which situations are related by events, we are able to characterize in a natural way the belief preconditions required for executing an action, and the effects of actions on the subsequent belief state of an agent. The interaction of observation and reasoning about situations gives an agent the power to plan actions that perform tests, as well as change the state of the world.

3.3.5 Formalizing Agents' Reasoning about Events

We now give a formalization that implements the ideas just laid out. The first requirement is that we be able to describe an agent a in situation s reasoning about other situations, especially the one just preceding. Since the formulas of $th(a, s)$ all refer to properties of situation s , we must enrich the OL so that formulas in the OL can refer to different situations. Using the techniques of belief-nesting of the previous section, we add to the OL the predicate H corresponding to the ML predicate of the same name. Then the OL expression $H(S_1, \lceil P \rceil)$ means that the OL' formula P holds in situation S_1 , regardless of what theory this formula appears in.¹³ With the addition of the H predicate to the OL, the notion that all formulas in $th(a, s)$ refer to properties of s can be formalized as:

$$\forall s f PR(th(a, s), f) \equiv PR(th(a, s), \lceil H(s, f) \rceil) \quad (H2)$$

H2 can be paraphrased by saying that an agent believes P in situation s just in case he

¹³ We will take $\{S_0, S_1, \dots\}$ to be standard names for situations in all languages. It will be assumed that standard names are always used to name situations.

believes that P holds in situation s . Given $H2$, it is possible to describe agents' theories as consisting purely of formulas in H ; but the added level of embedding puts this technique at a disadvantage with respect to using other predicates from OL to describe an agent's beliefs about the current situation.

It is also possible to formalize the notion that beliefs about previous situations persist, or are carried over into succeeding situations. Suppose that in situation S_n an agent has a belief of the form, "in a previous situation S_i , P was true." Then if S_{n+1} is the successor to S_n under some event, this belief is still valid. Formally, we can assert this with the ML axiom:

$$\forall s_i s_f e \text{ EV}(e, s_i, s_f) \supset [\forall a s f B(a, s_i, \lceil H(s, f) \rceil) \supset B(a, s_f, \lceil H(s, f) \rceil)] \quad (H3)$$

The antecedent of the implication says that s_i and s_f must be connected by some event for beliefs to be carried over from s_i to s_f ; this is necessary because we don't want agents to inherit beliefs from their future states. By phrasing the beliefs in terms of the predicate H , $H3$ carries over beliefs about all situations previous to and including s_i .

One of the consequences of $H3$ is that once an agent forms a belief about a situation, he holds that belief about that situation for all time. Since beliefs can be mistaken, it might happen that an agent observes something that forces him to revise his previously held beliefs. In that case, $H3$ is too strong, and the resultant theory will be inconsistent. We recognize that the general problem of reconciling inconsistent beliefs that arise from different sources (called *belief revision*) is a hard one, involving both conceptual and technical issues, and it is not part of this research to say anything new about it.¹⁴ Nevertheless, it is worthwhile to note that because the ML has terms that refer to agents' theories in different situations, it may be possible to describe a belief revision process formally in the ML.

¹⁴ Doyle [Doy78] worked on this problem under the rubric "Truth Maintenance," and more recent work in non-monotonic reasoning also considers this problem.

3.3.6 An Example of a Test

Given the preceding techniques for describing what an agent believes to hold in situations other than the one he is currently in, we can show formally that A_0 can use the *LT* axioms as a test to figure out whether the pilot light is on or not. In the initial situation S_0 , we will assume that A_0 knows he is at the oven O (where O is the standard name for the oven), and realizes that it is not lit:

Initial Conditions in the ML

- (1) $K(A_0, S_0, \lceil AT(A_0, O) \wedge \sim LIT(O) \rceil)$ given
- (2) $K(A_0, S_1, \lceil EV(do(A_0, light(O)), S_0, S_1) \rceil)$ given

The style of proof we will exhibit will be natural deduction, with assumption dependencies noted in square brackets in the justification for a line of the proof. Given the initial conditions, we next show that A_0 can observe whether or not the oven is lit in situation S_1 :

- (3) $\forall f SAF(f) \wedge f \neq \lceil LIT(O) \rceil \supset H(S_0, f) \equiv H(S_1, f)$ 2,B2,LT2
- (4) $SAF(\lceil AT(A_0, O) \rceil)$ definition of *SAF*
- (5) $H(S_1, \lceil AT(A_0, O) \rceil)$ 1,3,4,B2
- (6) $K(A_0, S_1, \lceil LIT(O) \rceil) \vee K(A_0, S_1, \lceil \sim LIT(O) \rceil)$ 5,O1

Line 3 comes from the frame axiom for *light*, and lets us infer that A_0 is still at the oven in situation S_1 (line 5). The observation axiom *O1* is then invoked to assert that A_0 will know what the state of the oven is in that situation.

Throughout this proof, we will be interested in two theories of the OL: $ths(A_0, S_0)$ and $ths(A_0, S_1)$. Assertions in the ML involving A_0 's beliefs can be reasoned about by using

semantic attachment to the appropriate OL theory. For example, line 1 above is attached to the following statements in $ths(A_0, S_0)$:

A_0 's Theory in Situation S_0

- (7) $AT(A_0, O) \wedge \sim LIT(O)$ 1, B2, semantic attachment
 (8) $H(S_0, \lceil \sim LIT(O) \rceil)$ 1, B2, H2, semantic attachment

Line 7 is the attachment of line 1 to A_0 's theory in S_0 . Line 8 is derived from line 1 by the use of H2; it is useful because it will persist as a belief in the successor situation S_1 . Generally, beliefs that an agent derives about the current situation can be inherited into succeeding situations by expressing these beliefs with the H predicate.

At this point we do reasoning by cases. First assume the right disjunct of line 6; then for A_0 's beliefs in situation S_1 we have:

A_0 's Theory in Situation S_1

- (9) $\sim LIT(O)$ [9]:assumed, semantic attachment
 (10) $\sim H(S_1, \lceil LIT(O) \rceil)$ [9]:9, H2 semantic attachment
 (11) $EV(do(A_0, light(O)), S_0, S_1)$ 2, semantic attachment
 (12) $H(S_0, \lceil PL(O) \rceil) \supset H(S_1, \lceil LIT(O) \rceil)$ 11, LT1
 (13) $\sim H(S_0, \lceil PL(O) \rceil)$ [9]:10, 12 contrapositive
 (14) $H(S_0, \lceil \sim PL(O) \rceil)$ [9]:13, TR for H

The first part of the result is derived by line 14, namely, that if A_0 observes that the O is not lit in situation S_0 , then he knows that the pilot light was not on in situation S_0 . This sequence of steps is interesting because it illustrates the intermixture of proof techniques in the ML and OL. Lines 9, 10, and 11 come from statements in the ML about $ths(A_0, S_1)$. Line 10 is derived from line 9 in the ML by the application of axiom H2. Line 11 says that

A_0 believes that S_1 is the result of the *light(O)* action occurring in S_0 , and follows directly from line 2 and semantic attachment. Line 12 follows from line 11 and the event axiom *LT1*; it is assumed that A_0 believes this axiom. Finally, 13 and 14 follow, given that the truth-recursion axioms for H are made available in all theories in the OL.

The left disjunct of line 6 can be reasoned about in the following way (since lines 11 and 12 did not involve any assumptions, they can be used in this part of the proof also):

A_0 's Theory in Situation S_1

- | | |
|--|--|
| (15) $LIT(O)$ | [15]:assumed, sem. att. |
| (16) $H(S_1, \lceil LIT(O) \rceil)$ | [15]:15, <i>H2</i> , sem. att. |
| (17) $\sim H(S_0, \lceil LIT(O) \rceil)$ | 8, <i>H3</i> , <i>TR</i> for H , sem. att. |
| (18) $\sim [H(S_1, \lceil LIT(O) \rceil) \equiv H(S_0, \lceil LIT(O) \rceil)]$ | [15]:16, 17 |
| (19) $H(S_0, \lceil \sim PL(O) \rceil) \supset H(S_1, \lceil LIT(O) \rceil) \equiv H(S_0, \lceil LIT(O) \rceil)$ | 11, <i>LT1</i> |
| (20) $\sim H(S_0, \lceil \sim PL(O) \rceil)$ | [15]:18, 19 contrapositive |
| (21) $H(S_0, \lceil PL(O) \rceil)$ | [15]:20, <i>TR</i> for H |

Here again, the first few lines (15, 16 and 17) are established by reasoning at the ML about $th_s(A_0, S_1)$. Line 17 comes from an instance of axiom *H3*, which enables an agent's beliefs to persist through a sequence of situations. Line 19 comes from A_0 's knowledge of *LT1*, and line 20 is the key step: it establishes that under the assumption of O being lit in S_1 , the pilot light was on in S_0 . Finally, the frame axiom *LT2* will carry the pilot light's status in S_0 forward into S_1 :

A_0 's Theory of Situation S_1

- (22) $\forall f \text{ SAF}(f) \wedge f \neq \text{LIT}(O) \supset H(S_0, f) \equiv H(S_1, f)$ 11,LT1
 (23) $\text{SAF}(\text{PL}(O))$ definition of SAF
 (24) $H(S_0, \text{PL}(O)) \equiv H(S_1, \text{PL}(O))$ 22,23
 (25) $\text{PL}(O)$ [15]:21,24,H2
 (26) $\sim \text{PL}(O)$ [9]:14,24,H2

Line 25 is under the assumption of the left disjunct of line 6, and line 26 is under the right disjunct. In the ML we can derive several results from the preceding proof structure:

In the ML

- (27) $B(A_0, S_1, \text{PL}(O)) \vee B(A_0, S_1, \sim \text{PL}(O))$ 6,20,21
 (28) $B(A_0, S_1, \text{LIT}(O)) \supset B(A_0, S_1, \text{PL}(O))$ 15,25
 (29) $B(A_0, S_1, \sim \text{LIT}(O)) \supset B(A_0, S_1, \sim \text{PL}(O))$ 9,26

Line 27 says that in S_1 , A_0 will either believe that the pilot light is on, or he will believe that is not on. Thus, by performing the action of lighting the oven, A_0 gains knowledge about the state of an unobservable, the pilot light. This is the desired result of agent A_0 using $LT1$ to perform a test of an unobservable property.

Lines 28 and 29 give proof-theoretic analogues to the LT axioms, which described the event of lighting the oven solely in terms of the actual situations before and after the event. These assertions show how the beliefs of A_0 change under the influence of the event $do(a, \text{light}(O))$. By suitably generalizing the preceding proof, it can be shown that 28 and 29 hold for all agents and initial situations:

$$\begin{aligned} \forall a, s_i, s_f \text{ EV}(do(a, \text{light}(o)), s_i, s_f) \wedge K(a, s_i, \text{AT}(a, o) \wedge \sim \text{LIT}(o)) \supset \\ B(a, s_f, \text{LIT}(o)) \supset B(a, s_f, \text{PL}(o)) \\ B(a, s_f, \sim \text{LIT}(o)) \supset B(a, s_f, \sim \text{PL}(o)) \end{aligned} \quad (LT3)$$

LT3 is valid under the condition that *LT1* is assumed to be believed by all agents. *LT3* is one description of the way in which an agent's beliefs change in a situation that results from an oven-lighting event; it would be most useful to a planner as a lemma to be invoked if the state of the pilot light were to be tested as a step in a plan. Another lemma about oven-lighting that would be useful to a planner would be one in which the belief preconditions to an action were made explicit; this would be used to plan actions that light the oven.

3.3.7 Plans and Planning

In the previous subsection we saw how to characterize the changes to an agent's beliefs produced by his observations of events. In this subsection we will consider how to use these results as part of the deductions that an agent needs to do to construct workable plans, i.e., plans that will accomplish their goals.

Consider how an agent might go about constructing workable plans. Using his description of various events (*PO*, *LT*, and others) he can try to find a sequence of actions that lead to the desired goals being true in some final situation. If we identify the planning agent with the ML, then a plan would be a sequence of situations connected by actions performed by that agent, such that the goals are true in the final situation. This doesn't seem to involve the planning agent in any reasoning about his beliefs; all he needs to do is describe how the actual world changes under the influence of his actions.

This isn't the whole story, though. The plan that is derived must be an *executable* plan; that is, if the plan is a sequence of actions, the agent must be able to execute each of those actions at the requisite time. For instance, the action description *light(oven(John))* will not be executable if *A₀* doesn't know which oven is John's. For a plan to be executable by an agent, the agent must know what action is referred to by each of the *do*-terms in the plan. According to a previous section, this means that the agent must have the standard name for the action in his theory. But what are standard names for actions? Following Moore [Moo80], we take the viewpoint that actions can be analyzed as a general procedure

applied to particular arguments, e.g., *puton* is a general procedure for putting one block on top of another, and *puton*(*A*, *B*) is that procedure applied to the two blocks *A* and *B*. If we assume that all agents know what general procedure each action denotes, then the standard names for actions are simply the terms formed by the action function applied to the standard names of its parameters.¹⁵ The condition that actions be executable forces the planning agent to make the critical distinction between his beliefs at planning time and his beliefs at execution time. A planning agent may not know, as he forms his plan, exactly what action a particular *do*-term in his plan denotes; but if he can show that at the time he is to execute that action, he will know what it is, then the plan is an executable one. Plans of this type occur frequently in common-sense reasoning; consider a typical plan A_0 might form to tell someone what time it is. The plan has two steps: first A_0 will look at his watch to find out what the time is, and then he will communicate this information to the requester. At planning time, A_0 doesn't really know what the second action is, because he doesn't know the time, and the time is an important parameter of the communication act. Yet he can reason that after looking at his watch, he will know the time; and so the plan is a valid one.

By this argument, an agent must analyze at planning time what the future states of his beliefs will be as he executes the plan. Thus the planning process intrinsically forces the agent into introspection about his future beliefs. Since we have identified the planning agent with the ML, it is natural to represent his future beliefs during the execution of the plan as OL theories in the situations that the planning process gives rise to. If the planning agent is A_0 , then these theories are $ths(A_0, S_0)$ (the initial situation), $ths(A_0, S_1)$, etc., where each of the S_i results from its predecessor via the execution of the next action

¹⁵ Actually, the condition that the parameters be standard names is too strong. Standard names have the property that every agent knows whether two individuals named by standard names are the same or not in every situation, but this condition is not strictly necessary for an action to be executable. Consider the action of requesting information from the telephone operator; surely it is not required that an agent be able to differentiate the operator from every other individual in his beliefs. If he were to dial the operator on two separate occasions, he would not necessarily be able to tell if he talked to the same operator or not.

in the plan. A_0 's planning process is basically a simulation of the plan's execution in which he reasons about the changes that both the actual world and his set of beliefs will undergo during the course of the plan's execution. By figuring out what his future states of belief will be, he can decide at planning time whether an action of the plan will be executable.

For A_0 to take other agents' plans into account in forming his own, he must be able to represent their future states of belief, in addition to his own. But this doesn't involve any additional representational complexity, since A_0 is already keeping track of his own beliefs during the simulated execution of the plan. In [Kon80] an example of a multiagent plan is presented; currently we are working on formalizing such plans in the framework presented here.

Actually, this planning process bears a strong resemblance to typical implementations of a situation calculus approach to planning [War74]. In these systems, events are axiomatized along the lines of *PO* and *LT*, and the planner searches for a sequence of situations that leads to the goal by doing theorem-proving with the event axioms; the search space is essentially the same in either approach. The main difference is in the relative complexity of reasoning that the two planning systems must be able to handle. In the approach described here, the effect of actions on the agent's beliefs in each situation greatly increases the deductive complexity of the planner and the work that it must do at each node in the search space of plans. The usefulness of lemmas such as *LT3* that describe the effects of actions on an agent's belief state now becomes apparent: by summarizing the effect of actions on an agent's beliefs, they reduce the complexity of the deductions that must be performed at each step in the plan. Further savings can be realized by using the method of belief attachment described in the previous section: from $H(s, f)$ at the ML, infer $K(A_0, s, f)$. Most of the work of figuring out A_0 's future states of knowledge can be performed by reasoning about H at the metalevel, rather than K , and this is considerably simpler. Finally, it should be noted that the executability requirement acts as a filter on plans. Thus a reasonable search strategy would be to first find a plan that works without

taking into account its executability (and hence the future belief states of the planning agent), and then test it for executability.

3.3.8 Conclusion

To summarize the contributions of this paper: we have defined a syntactic approach to the representation of knowledge and belief in which the key element is the identification of beliefs with provable expressions in a theory of the object language. The technique of *semantic attachment* to the intended interpretation of the metalanguage provability predicate has been advanced as a method of simplifying proofs by directly modeling an agent's inference procedure, rather than simulating it.

To unify a formalization of knowledge and action, we have shown how to take Moore's account of their interaction and formalize it within the syntactic framework. The benchmark example was a presentation of a test in which an agent uses his knowledge of observable properties of the world and the way actions affect the world to discover the state of an unobservable property. Finally, we pointed out how the formalization could be used in a planning system.

While this paper is a step towards showing that the syntactic approach can be extended to an adequate formalization of the interaction of knowledge and action, there is still much work to be done in constructing a practical planner for a multiagent environment that uses this formalism. Two areas in particular are critical. First, a suitable system for doing automatic deduction in the framework has to be worked out. Although we have advocated semantic attachment as a means of simplifying proofs, we have not yet explored the problem of controlling a deduction mechanism that uses this technique. The second area also involves control issues: how can a planner be designed to search the space of multiagent possible plans efficiently? One of the ideas suggested by this paper is to derive lemmas of the form of *LT3* that show the effect of actions on an agent's beliefs. With such lemmas, a planning system would have already compiled the necessary results for

constructing new belief states from previous ones.

3.4. Acknowledgments

This research is a direct outgrowth of research conducted with Nils Nilsson [Kon80], and still reflects his influence. Stan Rosenschein and Nils Nilsson read previous drafts of this paper, and their criticisms and comments have contributed to the final form. Also, I have benefited from talks with Pat Hayes, Bob Moore, Richard Weyhrauch, Carolyn Talcott, and all the members of the planning group at the Artificial Intelligence Center at SRI. This research is supported by the Office of Naval Research under Contract No. N00014-80-C-0296.

Appendix A: The Wise Man Puzzle

This is a solution to a simple version of the wise man puzzle, for whose statement we quote from [McC80]: *A king wishing to know which of his three wise men is the wisest, paints white dots on each of their foreheads, tells them that at least one spot is white, and asks each to determine the color of his spot. After a while the smartest announces that his spot is white reasoning as follows: "Suppose my spot were black. The second wisest of us would then see a black and a white and would reason that if his spot were black, the dumbest would see two black spots and would conclude that his spot is white on the basis of the king's assurance. He would have announced it by now, so my spot must be white."*

We will simplify this puzzle by having the king ask each wise man in turn what color his spot is, starting with the dumbest. The first two say "no," and the last says that his spot is white. Note that in this formalization we will not prove that the first two wise men don't know the color of their spots, but will take it as given; the deduction of such forms of non-knowledge is a hard problem that is not addressed here.

In formalizing the puzzle, we will take the three wise men to be A_0 , A_1 , and A_2 , in order of increasing stupidity. We will reason about the puzzle from A_0 's point of view, and show that A_0 knows that his spot is white after hearing the replies of the other two. We will not be concerned with the axiomatization of the speech act performed by the agents; it will be assumed that A_0 's model of the world changes appropriately to reflect this new information.

There are three situations in the puzzle: the initial situation S_0 , the situation S_1 just after A_2 speaks, and the situation S_2 just after A_1 speaks. The frame axioms for these situations are simply that every agent knows what he knew in the previous situation; these frame axioms are common knowledge.

We will identify A_0 with the ML, so the goal is to show:

$$H(S_2, W(A_0))$$

in the ML. $W(a)$ is the predicate whose meaning is "a's spot is white." The initial conditions of the problem are:

- (1) $W(A_1) \wedge W(A_2)$
- (2) $CFACT(\neg W(A_0) \vee W(A_1) \vee W(A_2))$
- (3) $CFACT(\neg K(A_2, S_0, \neg W(A_0)) \vee K(A_2, S_0, \neg \neg W(A_0)))$
- (4) $CFACT(\neg K(A_2, S_0, \neg W(A_1)) \vee K(A_2, S_0, \neg \neg W(A_1)))$
- (5) $K(A_1, S_0, \neg W(A_0)) \vee K(A_1, S_0, \neg \neg W(A_0))$
- (6) $CFACT(\neg \neg K(A_2, S_0, \neg W(A_2)))$
- (7) $CFACT(\neg \neg K(A_1, S_1, \neg W(A_1)))$

Line 1 says that A_0 observes white spots on A_1 and A_2 ; line 2 asserts that it is common knowledge that at least one spot is white. The next two lines state that it is common knowledge that A_2 can observe whether the other two agent's spots are white or not. Line 5 says that A_1 knows the color of A_0 's spot. And the last two lines express the effect of the first two agent's answers to the king on everyone's knowledge. This axiomatization will be sufficient to prove that A_0 knows his spot is white in S_2 .

The first step in the proof is to show that A_1 knows, in situation S_1 , that either his own or A_0 's spot is white; this by reasoning about A_2 's answer to the king. We will attach to A_1 's theory in situation S_1 (that is, $ths(A_1, S_1)$), and do our reasoning there:

A_1 's Theory in Situation S_1

- | | |
|--|------------------------|
| (8) $\neg K(A_2, S_0, \neg W(A_2))$ | 6, semantic attachment |
| (9) $K(A_2, S_0, \neg W(A_0) \vee W(A_1) \vee W(A_2))$ | 3, semantic attachment |
| (10) $K(A_2, S_0, \neg (\neg W(A_0) \wedge \neg W(A_1)) \supset W(A_2))$ | 9 |
| (11) $K(A_2, S_0, \neg \neg W(A_0) \wedge \neg W(A_1) \supset K(A_2, S_0, \neg W(A_2)))$ | 10, MP |
| (12) $\neg K(A_2, S_0, \neg \neg W(A_0) \wedge \neg W(A_1))$ | 8, 11 contrapositive |

In these lines, we have used the fact that everyone knows that everyone knows common knowledge assertions. At line 12, A_1 realizes that A_2 doesn't know that both A_0 and A_1 lack white dots; if he did, he would have announced the fact.

Now A_1 uses the common knowledge that A_2 can observe the color of A_0 's and A_1 's dots to reason that one of the latter has a white dot:

A_1 's Theory in Situation S_1

- | | | |
|------|--|-------------------------|
| (13) | $K(A_2, S_0, \lceil \sim W(A_0) \rceil) \wedge K(A_2, S_0, \lceil \sim W(A_1) \rceil)$ | [13]:assumption |
| (14) | $K(A_2, S_0, \lceil \sim W(A_0) \wedge \sim W(A_1) \rceil)$ | [13]:13,PR |
| (15) | $\sim K(A_2, S_0, \lceil \sim W(A_0) \rceil) \vee \sim K(A_2, S_0, \lceil \sim W(A_1) \rceil)$ | 13; 12,14 contradiction |
| (16) | $\sim K(A_2, S_0, \lceil \sim W(A_0) \rceil)$ | [16]:assumption |
| (17) | $K(A_2, S_0, \lceil W(A_0) \rceil) \vee K(A_2, S_0, \lceil \sim W(A_0) \rceil)$ | 3,semantic attachment |
| (18) | $K(A_2, S_0, \lceil W(A_0) \rceil)$ | [16]:16,17 |
| (19) | $\sim K(A_2, S_0, \lceil \sim W(A_1) \rceil)$ | [19]:assumption |
| (20) | $K(A_2, S_0, \lceil W(A_1) \rceil) \vee K(A_2, S_0, \lceil \sim W(A_1) \rceil)$ | 4,semantic attachment |
| (21) | $K(A_2, S_0, \lceil W(A_1) \rceil)$ | [19]:19,20 |
| (22) | $K(A_2, S_0, \lceil W(A_0) \rceil) \vee K(A_2, S_0, \lceil W(A_1) \rceil)$ | 15,16,18,19,21 |
| (23) | $H(S_0, \lceil W(A_0) \vee W(A_1) \rceil)$ | 22,B2 |
| (24) | $W(A_0) \vee W(A_1)$ | 23,frame axioms,R1 |

We first show here that A_2 doesn't know A_0 's spot is black, or he doesn't know that A_1 's spot is black (line 15). Assertions that follow from assumptions are indicated by a square bracketing of the assumption line number in their justification. Next we do an analysis by cases of line 15; in either case, line 22 holds: A_2 either knows A_0 's spot is white, or he knows A_1 's spot is white. From this A_1 concludes that either he or A_0 has a white spot (line 24). Note that the frame axioms were needed to show that the W predicate doesn't change from situation S_0 to situation S_1 .

At this point we are through analyzing A_1 's theory of situation S_1 , and go back to the ML to reason about situation S_2 . By line 5, A_1 knows the color of A_0 's dot, so we assume that he knows it is black:

At the Metalevel

- | | | |
|------|---|-------------------------|
| (25) | $K(A_1, S_0, \lceil \sim W(A_0) \rceil)$ | [25]:assumption |
| (26) | $K(A_1, S_1, \lceil \sim W(A_0) \rceil)$ | [25]:25,frame axioms |
| (27) | $K(A_1, S_1, \lceil \sim W(A_0) \supset W(A_1) \rceil)$ | 24,frame axioms |
| (28) | $K(A_1, S_1, \lceil W(A_1) \rceil)$ | [25]:26,27,MP |
| (29) | $\sim K(A_1, S_1, \lceil W(A_1) \rceil)$ | 7, common knowledge |
| (30) | $\sim K(A_1, S_0, \lceil \sim W(A_0) \rceil)$ | 25, 28,29,contradiction |
| (31) | $K(A_1, S_0, \lceil W(A_0) \rceil)$ | 5,30 |
| (32) | $H(S_0, \lceil W(A_0) \rceil)$ | 31,B2 |

Under the assumption that A_1 knows A_0 's spot is black, we derive the contradiction of lines 28 and 29. Therefore, by line 5, it must be the case that A_1 knows A_0 's spot to be white. This is the conclusion of line 32; since this is one of A_0 's beliefs, we are done.

4. Planning Natural-Language Utterances

This section was written by Douglas Appelt. This research was supported in part by the Office of Naval Research under contract N0014-80-C-0296 and in part by the National Science Foundation under grant MCS-8115105.

4.1. Introduction

This paper describes recent research on a natural-language generation system that is based on planning. The view of language production adopted here is similar to that of Allen [All80], Cohen and Perrault [Coh79], namely that speakers produce utterances with the intention of satisfying particular goals, and that a hearer's understanding of an utterance depends on how he interprets the utterance as fitting in with what he believes to be the speaker's plan.

A system named KAMP (for Knowledge And Modalities Planner) has been developed that plans natural-language utterances, starting with a high-level description of the speaker's goals. The system can be viewed as an extension of speech-act planning research by Cohen [Coh79], but while Cohen stopped at producing abstract descriptions of speech-acts, KAMP extends the planning down to the level of the production of English sentences, integrating both physical and linguistic actions that satisfy discourse, knowledge-state and referring goals into utterances that achieve multiple goals simultaneously.

This research has addressed the following three major problems:

- Developing a domain-independent multiple-agent planning system called KAMP.
- Extending the possible-worlds semantics representation of propositional attitudes developed by Moore [Moo80] to handle mutual knowledge and wanting.

Figure 1

Satisfying Multiple Goals with a Request

- Capturing linguistic knowledge in the axioms, critics, and procedures used by KAMP to facilitate the planning of *linguistic* actions.

4.2. Why Plan Utterances?

Figure 1 illustrates a typical situation arising when two people cooperate on a common task in which a speaker plans an utterance that has multiple effects on the intended hearer. The speaker points to one of the tools on the table and says "Use the wheelpuller to remove the flywheel." The hearer, who is observing the speaker while he makes the request, and knows that the speaker is pointing to the particular tool, thinks to himself, "Ah, so that's a wheelpuller. I was wondering how I was going to get the flywheel off."

The speaker's utterance in figure 1 is syntactically very simple, but a surprising amount of sophisticated reasoning is required for a speaker to produce such an utterance and know that it will have the intended effect on the hearer. Most obviously, the speaker wants to make a request of the hearer to do something. However, before he can make the request, he has to determine whether the hearer has enough knowledge to carry it out. If not, then the speaker has to know that the hearer can form a plan for acquiring the knowledge, or he must furnish the knowledge himself. In this example, the speaker *informs* the hearer that he should use the wheelpuller as part of the same utterance that he uses to request that he perform a removing action.

The speaker uses the noun phrase "the wheelpuller" to refer to a particular tool. In figure 1, it is evident from the hearer's reaction that he didn't know what a wheelpuller is, and the speaker knew that, because he performed a pointing action to make his intention to refer clear. Although the speaker knew that the hearer did not know what a wheelpuller was, he knew that the hearer would know after understanding the sentence. The utterance of figure 1 also serves to inform the hearer that the object the speaker is pointing to is a wheelpuller. In order for the speaker to make that inference, he had to know that the hearer would know that he didn't intend the object he was pointing at to be the referent of "the flywheel." He knows that because he knows the hearer knows that the flywheel is not a tool, and therefore cannot fill the instrument role of "remove." Under different circumstances, the speaker could point to the flywheel, utter the identical sentence as in figure 1, and reason that the utterance would inform the hearer that the object he was pointing to was the flywheel.

4.3. The KAMP Language Planning System

It is clear from the above example that a model of language production that is simply a transducer from a logical form to a surface utterance is not sufficient to account for the way that people use utterances to satisfy multiple goals — an ability that requires the speaker and hearer to make inferences about each other's plans. Furthermore, utterances do more than alter

Figure 2

The Organization of a Language Planning System

the participants' knowledge and wants. They influence the participants' emotional attitudes, and affect the state of the ongoing discourse. Utterances can be planned with intentions of achieving goals along these dimensions as well. Therefore, instead of a simple transducer from logical form, KAMP is organized like the planner in figure 2.

The overall organization of KAMP as a hierarchical planner similar in overall organization to Sacerdoti's NOAH [Sac77] was discussed in detail in Appelt [App80] [App82]. KAMP has two descriptions of actions at each level in the action hierarchy: A full axiomitization in terms of possible worlds, and a shorter, more intuitive description called an *action summary*. KAMP uses the action summaries as a heuristic device to propose plans that it then verifies using the

Figure 3
A Hierarchy of Actions Related to Language

possible worlds axiomatization. The heuristic plan generation process is implemented by the NOAH-like hierarchical planner, while the verification process is implemented by a first-order-logic theorem-prover.

Figure 3 illustrates the hierarchy of actions that is used by KAMP to plan linguistic actions. The central problem of building a language-planning system around KAMP is formulating the correct axioms, and incorporating the correct action summaries and critic procedures into KAMP that describe the actions of the hierarchy in Figure 3.

4.4. Axiomatizing Knowledge about Intensional Concepts

Axiomatizing the actions of Figure 3 requires the ability to specify how performing actions affects the knowledge, mutual knowledge and wants of agents. Moore's possible-worlds-semantics approach [Moo80] solves this problem with respect to knowledge and its relation to action. It is impossible to describe Moore's approach in detail here, but the central idea is to axiomatize the possible-worlds semantics of a modal object language in a first-order meta-language. Thus, the semantics of a statement like $\text{Know}(A, P)$ is represented as " P is true in every possible world consistent with what A knows."

It is necessary to reason about mutual knowledge (i.e., knowledge that A knows that B knows that A knows ... ad infinitum) to plan referring expressions (see Clark & Marshall, [Cla81]). KAMP reasons about the mutual knowledge shared by two agents by reasoning about what is true in the union of the sets of possible worlds consistent with two agents' knowledge. An "agent" called the *kernel* of A and B is defined, for whom the worlds consistent with his knowledge are precisely that union. This is a generalization of the "any fool" approach advocated by McCarthy and Sato [McC78].

Wanting is represented in KAMP by a relation between an agent and a set of possible worlds called a *situation*. The situation is a set of possible alternatives to the current world which an agent is said to want. The situation that an agent wants can be characterized by different propositions according to what he knows. An agent is said to want P if there is some situation he wants such that P is true in every possible world that is a member of the situation, with the terms of P evaluated with respect to the agent's knowledge. This representation allows one to make a connection between knowledge and wanting, which while ignoring many of the subtle problems associated with wanting and intention, is adequate for solving many planning problems that arise in the task-oriented domains under consideration. It allows one to reason, for example, that if John wants to meet the president of the United States, and if John knows that the president of the United States is Ronald Reagan, then John wants to meet Ronald

Reagan.

4.5. Axiomatizing Linguistic Actions

As illustrated in figure 3, the most abstract linguistic actions in KAMP's hierarchy are *illocutionary acts*. These are actions such as informing, requesting, promising, thanking, etc., that can be characterized as communicative acts independent of any particular linguistic realization.

Speakers do not perform illocutionary acts directly, but rather perform them *by means* of the performance of *surface speech acts*. When a speaker plans a surface speech-act, he selects the propositional content of the sentence he is going to utter (which may be different from the propositional content of the illocutionary act in the case of indirect speech acts), and selects a particular syntactic structure that is to be used for the realization of the illocutionary act. A fundamental choice made at this level is whether to use an imperative, interrogative or declarative sentence. Each surface speech-act has a syntactic structure tree associated with it that evolves as the plan is expanded to include more constituents expanded to progressively lower levels.

The relationship between illocutionary acts and surface speech-acts is similar to the relationship between walking across the room and a sequence of muscle movements. One action is performed by means of performing the others. What distinguishes this relationship from walking and muscle movements is that the particular illocutionary act that is being performed depends on the hearer's recognition of what the speaker is trying to do. A particular surface speech-act, for example, "Can you reach the tool on the top shelf?" can in one case be a request to retrieve a tool, and in another case a request to inform the speaker of the ability to retrieve a tool.

As the example illustrates, it is not even the case that there is a one-to-one correspondence between illocutionary acts and surface speech acts. In this case several informing and requesting actions are being performed as part of a single surface speech-act.

KAMP has an axiomatization of each illocutionary act and surface speech-act it knows about in terms of the possible worlds approach outlined above. This paper will not describe the axioms in detail, so the interested reader is referred to [App80] for more information. The general approach to axiomatizing illocutionary acts is to describe only what Austin [ref] refers to as illocutionary effects, and not perlocutionary effects. In other words, the effect of informing a hearer that *P* is not that the hearer then believes *P*, but that the hearer knows that the speaker wants him to know that the speaker believes *P*. However, the speaker wanting the hearer to know that the speaker believes *P* is a reasonable *precondition* of the sincere performance of an informing action. Therefore, the effects of an illocutionary act can be stated as producing the mutual knowledge between the speaker and the hearer that the act has been performed. All deductions about the change in the knowledge of the participants follow from knowing the action has been performed, and their mutual knowledge of the conditions on the action's performance.

Surface speech-acts include *concept activation actions* as part of their realization on the next lower level of abstraction. Concept activation actions describe referring at a high enough level of abstraction so that they are not constrained to be purely linguistic actions. When a concept activation action is expanded to a lower level of abstraction, it can result in the planning of a noun phrase within the surface speech act of which the concept activation is a part, *and* physical actions such as pointing that also communicate the speaker's intention to refer, and may be realized by a plan that includes either physical or linguistic actions.

Although concept activation actions can be realized through physical actions, the planner must reason about their interaction with the linguistic actions being planned. Therefore, concept activation actions are expanded with an *intention-communication component* that communicates the speaker's intention to refer, and a *linguistic-realization component* that is part of the surface speech-act and takes into account the grammatical rules.

The lowest level actions of Figure 3 are the utterance acts. Utterance acts consist of the utterance of particular sequences of words. The component of KAMP that produces utterance

acts from a plan of hierarchical linguistic actions and the constituent-structure trees associated with surface speech acts is quite simple because at this point, no modifications to the plan are made because decisions are completely determined by grammatical rules, and there is no room for the speaker's intentions to influence the process. The final stage of planning consists primarily of making obligatory modifications required by the grammar, such as subject-verb agreement, proper auxiliary affixes, insertion of reflexive pronouns, etc.

4.6. Conclusion

The development of KAMP has been the first step toward a theory of planning natural language utterances that allows the satisfaction of multiple goals in a single surface utterance, that plans utterances tailored to the specific knowledge of an intended hearer as well as the context of the discourse, and that provides for the integration of physical and linguistic actions.

There are a number of areas in which the concepts developed in KAMP can be profitably applied and extended. One major area is the planning of extended discourse. Currently, KAMP plans only very simple dialogues. It may plan more than one utterance if it wants to perform several illocutionary acts, and it cannot figure out a way in which one can subsume the others. The resulting dialogues will be coherent because the illocutionary acts are naturally tied together by being part of the same plan. However, to move beyond simple dialogues consisting of alternating one or two sentence turns, more complex, abstract discourse-level actions must be defined. McKeown [McK80] incorporates such strategies in a language generation system, and such actions need to be formalized in a planning framework to be used by a system like KAMP.

KAMP currently keeps track of discourse focus primarily so it can generate appropriate referring expressions. When planning an extended discourse, the planner would also be concerned about the speaker's need to inform the hearer of topic shifts. Topic shifting actions, similar to those described by Reichman [Rei78], must be formalized and planned when appropriate.

The primary focus of research on KAMP has been on planning natural-language utterances. However, KAMP is a general tool that can serve as the basis of multiple-agent planning systems in a variety of domains. There are many problems concerning planning to acquire knowledge, cooperation among several agents with limited resources, etc., for which KAMP seems useful.

References

- [All80] Allen, J. and C. R. Perrault, *Analyzing Intention in Utterances*, **Artificial Intelligence**, 15:3 (December 1980).
- [App80] Appelt, D. E., *A Planner for Reasoning about Knowledge and Action*, **Proceedings of the First Annual National Conference on Artificial Intelligence**, Stanford, California (August 1980).
- [App82] Appelt, D. E., *Planning Natural Language Utterances to Satisfy Multiple Goals*, Technical Note 259, Artificial Intelligence Center, SRI International, Menlo Park, California (1982).
- [Aus62] Austin, J., *How to do things with Words*, Oxford, England: Oxford University Press, 1962.
- [Chu51] Church, A., *A Formulation of the Logic of Sense and Denotation*, in **Structure, Method and Meaning**, P. Henle et. al., eds., New York, New York: Liberal Arts Press, 1951.
- [Cla81] Clark, H., and C. Marshall, *Definite Reference and Mutual Knowledge*, in **Elements of Discourse Understanding** A. Joshi et. al., eds., Cambridge, England: Cambridge University Press, 1981.
- [Coh79] Cohen, P. and C. R. Perrault, *Elements of a Plan Based Theory of Speech Acts*, **Cognitive Science**, 3:3, pp. 177-212 (1979).
- [Cre79] Creary, L. G., *Propositional Attitudes: Fregean Representation and Simulative Reasoning*, **Proceedings of the 6th International Joint Conference on Artificial Intelligence**, Tokyo, Japan, pp. 176-181 (1979).
- [Dij75] Dijkstra, E. W., *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*, **Communications of the ACM**, 18:8, pp. 453-457 (August 1975).
- [Doy78] Doyle, J., *Truth Maintenance Systems for Problem Solving*, Memo AI-TR-419, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (1978).
- [Fik71] Fikes, R. E. and N. J. Nilsson, *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*, **Artificial Intelligence**, 2:3-4, pp. 189-208 (Winter 1971).
- [Fol80] Follett, R., *Automatic Program Synthesis*, Ph.D. dissertation, University of New South Wales, Australia (May 1980).
- [Har79] Harel, D., *First Order Dynamic Logic*, **Lecture Notes in Computer Science**, 68, New York, New York: Springer-Verlag, 1979.

- [Hay79] Hayes-Roth, B. and F. Hayes-Roth, S. Rosenschein, and S. Cammarata. *Modeling Planning as an Incremental, Opportunistic Process*, **Proceedings of 8th International Joint Conference on Artificial Intelligence**, Stanford University, Stanford, California, pp. 375-383 (August 1979).
- [Hoa69] Hoare, C.A.R., *An Axiomatic Basis for Computer Programming*, **Communications of the ACM**, 12:10, pp. 576-583 (October 1969).
- [Kap71] Kaplan, D., *Quantifying In*, in **Reference and Modality**, L. Linsky ed., pp. 112-144, London, England: Oxford University Press, 1971.
- [Kle67] Kleene, S. C., **Mathematical Logic**, New York, New York: John Wiley and Sons, 1967.
- [Kon80] Konolige, K. and N. Nilsson, *Multiple Agent Planning Systems*, **Proceedings of the First Annual National Conference on Artificial Intelligence**, Stanford, California, (August 1980).
- [Kon81] Konolige, K., in preparation (1981).
- [Kow79] Kowalski, R., **Logic for Problem Solving**, New York, New York: North-Holland Publishing Company, 1979.
- [Lit77] Litvintchouk, S. D. and V. R. Pratt, *A Proof-Checker for Dynamic Logic*, **Proceedings of 5th International Joint Conference on Artificial Intelligence**, Massachusetts Institute of Technology, Cambridge, Massachusetts, pp. 552-558 (August 1977).
- [Man77] Manna, Z. and R. Waldinger, **Studies in Automatic Programming Logic**, New York, New York: North-Holland Publishing Company, 1977.
- [McC62] McCarthy, J., *Towards a Mathematical Science of Computation Information Processing*, **Proceedings of the IFIPS Congress**, 62, pp. 21-28, New York, New York: North-Holland Publishing Company, 1962.
- [McC69] McCarthy, J. and Hayes, P. J., *Some philosophical Problems from the Standpoint of Artificial Intelligence* in **Machine Intelligence 4**, B. Meltzer and D. Michie eds., pp. 463-502, Edinburgh, Scotland: Edinburgh University Press, 1969.
- [McC78] McCarthy, J., M. Sato, T. Hayashi, and S. Igarashi, *On the Model Theory of Knowledge*, Memo AIM-312, Artificial Intelligence Laboratory, Stanford University, Stanford, California (1978).
- [McC79] McCarthy, J., *First Order Theories of Individual Concepts and Propositions*, in **Machine Intelligence 9**, J. E. Hayes and D. Michie eds., pp. 120-147, New York, New York: Halsted Press, 1979.
- [McK80] McKeown, K. R., *Generating Relevant Explanations: Natural Language Responses to Questions about Database Structure*, **Proceedings of the First Annual National Conference on Artificial Intelligence**, Stanford, California (August 1980).

- [Moo80] Moore, R. C., *Reasoning About Knowledge and Action*, Technical Note 191, Artificial Intelligence Center, SRI International, Menlo Park, California (1980).
- [Nil80] Nilsson, N., *Principles of Artificial Intelligence*, Palo Alto, California: Tioga Publishing Company, 1980.
- [Pra76] Pratt, V. R., *Semantical Considerations on Floyd-Hoare Logic*, *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science*, pp. 109-121 (October 1976).
- [Pra78a] Pratt, V. R., *A Near-Optimal Method for Reasoning about Action*, Technical Note MIT/LCS/TM-113, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts (September 1978).
- [Pra78b] Pratt, V. R., *Six Lectures on Dynamic Logic*, Technical Note MIT/LCS/TM-117, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts (December 1978).
- [Qui71] Quine, W.V.O., *Quantifiers and Propositional Attitudes*, in *Reference and Modality*, L. Linsky ed., pp. 101-111, London: Oxford University Press, 1971.
- [Rei78] Reichman, R., *Conversational Coherency*, *Cognitive Science*, 2:4 (1978).
- [Ros81] Rosenschein, S. J., *Hierarchical Planning: Implementation Considerations*, forthcoming.
- [Sac75] Sacerdoti, E. D., *The Nonlinear Nature of Plans*, *Proceedings of 4th International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, USSR, pp. 206-214 (September 1975).
- [Sac77] Sacerdoti, E. D., *A Structure for Plans and Behavior*, Amsterdam, Holland: Elsevier North-Holland, Inc., 1977.
- [Sus75] Sussman, G. J., *A Computer Model of Skill Acquisition*, New York, New York: American Elsevier, 1975.
- [Tat77] Tate, A., *Generating Project Networks*, *Proceedings of 5th International Joint Conference on Artificial Intelligence*, Massachusetts Institute of Technology, Cambridge, Massachusetts, pp. 888-893, (August 1977).
- [Tat77] Tate, A., *Project Planning Using a Hierarchic Non-Linear Planner*, D.A.I. Research Report No. 25 University of Edinburgh, Edinburgh, Scotland (1977).
- [Wal75] Waldinger, R., *Achieving Several Goals Simultaneously*, Technical Note 107, Artificial Intelligence Center, SRI International, Menlo Park, California (July 1975).
- [War74] Warren, D.H.D., *WARPLAN: A System for Generating Plans*, Memo 76, Department of Computational Logic, University of Edinburgh, Edinburgh, Scotland (July 1974).

- [War76] Warren, D.H.D., *Generating Conditional Plans and Programs*, Proceedings of Summer Conference on Artificial Intelligence and Simulation of Behavior, University of Edinburgh, Edinburgh, Scotland, pp. 344-354 (July 1976).
- [Wey80] Weyhrauch, R., *Prolegomena to a Theory of Mechanized Formal Reasoning*, *Artificial Intelligence* 13 (1980).

END

FILMED

9-83

DTIC