METALANGUAGE AND META-REASONING

Robert E. Filman
John Lamping
Fanya S. Montalvo

Computer Research Center
Hewlett-Packard Laboratories
Palo Alto, California 94304

Abstract

We describe three experiments using meta-language and meta-reasoning to solve problems involving belief, heuristics, and points of view. These experiments use the knowledge representation system SPHERE, built at Hewlett-Packard Laboratories and based on the ideas of FOL [20]. A key idea underlying our research is the segmentation of knowledge into contexts and the meta-linguistic treatment of these contexts as objects.

I INTRODUCTION

Recent research in Artificial Intelligence has focused on meta-knowledge [2,0,8,14]. Meta-knowledge has attracted attention because of an increasing awareness that much of reasoning IS about language and the structure of knowledge. Several representation systems have mechanisms for embodying meta-linguistic constructs, among them KRL [3], Prolog [4], LCF [5], MRS [9] and Omega [11]. Most of these systems use a quoting mechanism to distinguish language from meta-language. One system, FOL [20], separates language and meta-language into different contexts and makes language structures true objects to meta-contexts. This results in a more expressive and uniform system.

In this paper we summarize three examples of meta-theoretic represention and reasoning. We implemented these examples in the representation system SPHERE, an intellectual descendant of FOL. We describe SPHERE in the next section; for the moment it is sufficient to understand that SPHERE encapsulates knowledge in contexts. These contexts can manipulate other contexts; a context that manipulates another is a meta-context. The first example uses a "tree" of meta-contexts to represent beliefs about beliefs. We reason about the beliefs of a participant in a competitive bidding situation. The second example illustrates describing heuristics declaratively in met a-contexts. In this example, we describe the game of Mastermind to a context and describe heuristics for playing the game to its meta-context. The last example shows how contexts can present and manipulate multiple views of a single underlying object. The problem domain is that of using graphics effectively to gain an understanding of Rubik's Cube.

II SPHERE

Key features of SPHERE are: 1) All statements are relative to some context. 2) A context is a triple, composed of its own *language* (individuals, functions and relations), a set of *facta* (WFF's), and a partial model, the *simulation structure* (semantic attachments). Conceptually, a context is a self-contained theory of some domain. 3) The basic inference mechanism is simplification: the reduction of an expression (term or WFF) to a simpler form. These systems have two important simplification mechanisms: syntactic simplification and semantic simplification. *Syntactic simpiication* uses the facts (axioms) of a context as re-writing rules. Semantic *simplification* uses the values of computationally effective procedures and data structures to transform expressions.* The power of these systems arises from using the data structures that encode expressions and contexts as the semantic attachments to constants in *meta-contexts.* Thus, a constant in the language of one context (the meta-context) can be semantically attached to the data structure that encodes another context (the subordinate context). The meta-context can manipulate that constant as it does any other constant. We present examples of this meta/subordinate manipulation later in this paper.

Contexts have parallels in systems such as FRL [10], Viewpoints [18], CONNIVER [17], and QA4 [18]. Manipulating the syntax of language independently from the mechanisms that effect its semantics parallels work on 3-LISP [19]. SPHERE, like its ancestor FOL, cleanly defines and uniformly integrates these mechanisms in a mathematically rigorous system. We have also used SPHERE as the semantic base for a natural language understanding system [7] and for a system that unifies the relational and algebraic database calculi.

III BID

Our first example shows how to represent some aspects of the concept of belief with multiple contexts. Our strategy is to embody an agent's beliefs in a context: entering that context to reason *with* the agent's beliefs and treating that context as an object to reason about the agent's beliefs.

We view the world subjectively from the perspective of (the context of) P, the president of a paint company. He bids on contracts to supply paint. In deciding what to bid, he needs to consider the bids and bidding practices of his competitors, Q and R. We represent P's belief's about Q and R's beliefs as the constants *Q* and *R* in P's context. We attach contexts to these constants. These contexts resemble P's context in many respects, including representing competitors as constants with attached contexts.

Belief contexts all share a common language of paint manufacturing. Making paint requires possession of a *process.* Processes vary between companies. Each process uses several *ingredients,* for any given process, p, and ingredient, i, *quanty(p,i)* is the amount of i used in p. *Calcium-carbonate, alkyd-resin,* and *titanium-dioxide* are typical ingredients.

* More specifically, the attachment to a constant, relation or function (syntactic objects) in a context is a pair: the name of a context and the name of a constant in that context. Frequently, this named context is the LISP context. Procedurally, semantic simplification translates (through attachments) the terms of an expression into an expression in another context, runs the simplification mechanism in that context, and translates the result back to the original context. (This translation process may involve passing through several intermediate contexts.) Not all contexts have the same simplifies In particular, the simplification mechanism in the LISP context is the Lisp evaluator. Thus, in semantic simplification, the data structures that encode contexts and expressions often become the arguments for Lisp functions that manipulate contexts and expressions.

Every paint company president believes he can get each ingredient for a given cost *cost(i),* though costs may vary between companies. For each process, the *proceucott(p)* is the cost of making paint with that process: the scalar product of the quantity and cost of each ingredient.

Each company also has its own *competitors* and a *pricing-practice.* A *Pricing-practice* is a function of a company's costs and the lowest bid of its competitors. Each *pricing-practice* determines the company's bid. Each belief context has this general knowledge about paint processes, arithmetic and bidding.

A context P represents the state of P's beliefs. This context has an axiom representing P's pricing practice and axioms representing P's beliefs about his costs for commodities. In context P, constant *Q (R)* represents P's belief's about "the state of Q's (R's) beliefs". Each of these constants is attached to a context (contexts Q and R). Each of these contexts has the language of paint bidding. Q has an axiom for pricing-practice, representing what "P believes is Q's pricing-practice". It has axioms representing what "P believes Q believes are Q's commodity costs". And it has two constants, *P* and *R.* This second constant, *R,* is attached to a context (Qr) that represents P's beliefs about Q's beliefs about R.

Clearly, we can continue hypothesising contexts indefinitely, representing "P's beliefs about Q's beliefs about P's beliefs about Q's..." In practice, we stop at the fourth level. Figure 1 shows the resulting context structure. We have fifteen contexts representing beliefs, arranged in a tree.

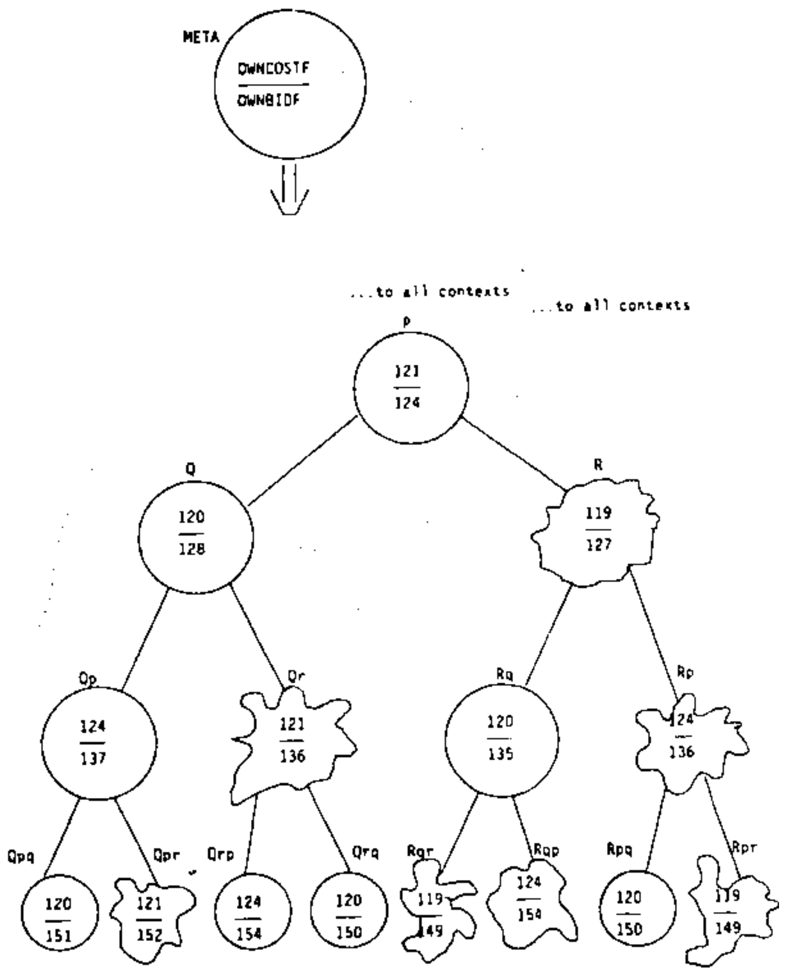

Figure 1. The context structure of BID. The top number in each context is context's cost (computed by the meta-context function OWNCOSTF), and the bottom number is the context's bid (computed by the meta-context function OWNBIDF). Contexts with normal pricing practices are circles; those with predatory pricing practices are blobs.

P's bid is the result of applying his pricing-practice to his minimum process cost and the lowest expected bid of his competitors. We use another meta-context to compute bids. The meta-context finds P's minimum cost, determines his competitors and computes the expected bid of each competitor in the competitor's context. The meta-context then determines the bid by simplifying the pricing-practice of the results. The computation is a recursion, similar to the familiar recursion of mini-maxing game trees. The recursion halts at the leaves of the context tree, where contexts do not have attachments to competitors. At that point, only a company's costs are used in computing its bid.

We used these structures to describe some "actual beliefs" to the system. Figure 1 summarizes the results of this test run. For the example, we asserted that P believes that 1) there is a standard process known to all manufacturers; 2) there is a proprietary process unique to P, but known to his competitors; 3) P has a new process, unknown to P's competitors; 4) Q has a sec ret-process, but that Q believes that P doesn't know about it; 5) everyone believes that Q can get cheaper *alkyd-resin,* 6) R can get cheaper *titanium-dioxide;* 7) Q doesn't know (0); 8) everyone knows that R is a predatory pricer while Q is not; and 9) each of Q and R view P's pricing practices to be like their own.

Using this data, our system calculates what P should bid. The example expresses (to a first approximation) some complicated relationships about the beliefs of others. This expression has the virtue of being both computationally effective and uniform.

Our use of multiple contexts to embody belief is in contrast to Appelt's use of possible world semantics for belief [1]. Possible worlds involve reasoning about the possible states of an agent, reducing, through the use of axioms and inference rules, this initially infinite set down to a solution set with the desired properties. Our context approach allows us to directly incorporate the knowledge of an agent in the same (axiomatic) format in which it was originally expressed and to constructively refine a single object (the context) as further inferences are made. Our approach implements some of the' ideas expressed by Konolige and Nilsson on multiple agent planning systems [15].

IV MASTERMIND

Our second example implements a program that plays Mastermind [13]. Meta-theory plays two key roles in this system. First, meta-Btatements express facts about the hidden sequence. Second, meta-meta-statements describe the control structure and heuristics that the player uses in deducing more information and in planning its next guess. We built a system with a subordinate context whose language describes guesses and solutions, a series of meta-contexts embodying successive knowledge refinements, and a meta-meta-context that describes control and heuristics.

The subordinate context's language has two kinds of objects, colors and positions, and a single "fundamental" predicate: *Positions P in the hidden sequence it color C.* This is sufficient to describe the hidden sequence. However, it is difficult to concisely express the information provided by guesses in the subordinate-context's language. Instead, we represent guesses as meta-context statements of the form "Exactly N of the following statements are true". Reasoning in Mastermind involves reasoning with these meta-Btatements *about* the fundamental predicate.

There are two useful types of meta-statements about the fundamental predicate. Imagine that the hidden sequence is <Red Blue Orange Green>. The guess <Red Blue Orange red> elicits the response that two colors are in the correct position and one is in an incorrect position. This information can be expressed as:

"Exactly two of the following:
 'The first position is red.',
 'The second position is blue.',
 'The third position is green.',
 'The fourth position is red.'
are true."
and
"Exactly three of the following:
 'There is at least one red.',
 'There are at least two reds.',
 'There is at least one blue.',
 'There is at least one green.'
are true."

These are both examples of a more general kind of statement — counting statements. Counting statements indicate the number of objects in a set that satisfy some predicate. In the present case, the predicate identifies objects attached to true statements in the subordinate context.

The meta-context has four axioms. Three allow a conclusion (from appropriate count information) of whether particular objects satisfy the predicate. The fourth relates the count of a set, A, and of its subset, B, to the count of the set difference A-B. These theorems (and a few others that relate counting to Mastermind) form the basis for all the necessary inferences.

A separate context, meta to the meta-context described above, contains the axioms that describe how to play the game. Figure 2 shows the context structure of the Mastermind player. The playing strategy is to discover a pattern consistent with what is known so far. The control strategy finds a consistent guess by making a hypothesis and inferring as much as it can. It repeats the "hypothesize and infer" cycle until it has restricted the possibilities to a single pattern. The system represents each hypothesis with a new context, built by extending a copy of the previous hypothesis-context with the new hypothesis. Hypotheses may turn out to be false, forcing the system to backtrack. The heuristic used to pick a hypothesis is to find a counting statement on the smallest set of sentences and hypothesize one of those sentences. When the hypotheses have restricted the possibilities to a single consistent pattern, the system guesses that pattern. The system repeats this guessing strategy until it discovers the secret pattern.

The inference part of the control cycle uses forward chaining. The inference mechanism is modified with a pruning heuristic and an ordering heuristic. The pruning heuristic discards less specific information implied by more specific information. For example, if I know the fact that two of the three colors red, blue, and green are present, and then discover that red is definitely present, I can conclude that either blue or green is present and then forget the original fact. The ordering heuristic suggests making inferences from more specific information before inferences from less specific information. The two heuristics change forward inferencing from a blind search to a procedure for improving the quality of information. We found these heuristics to be sufficient to focus the system on appropriate inferences.

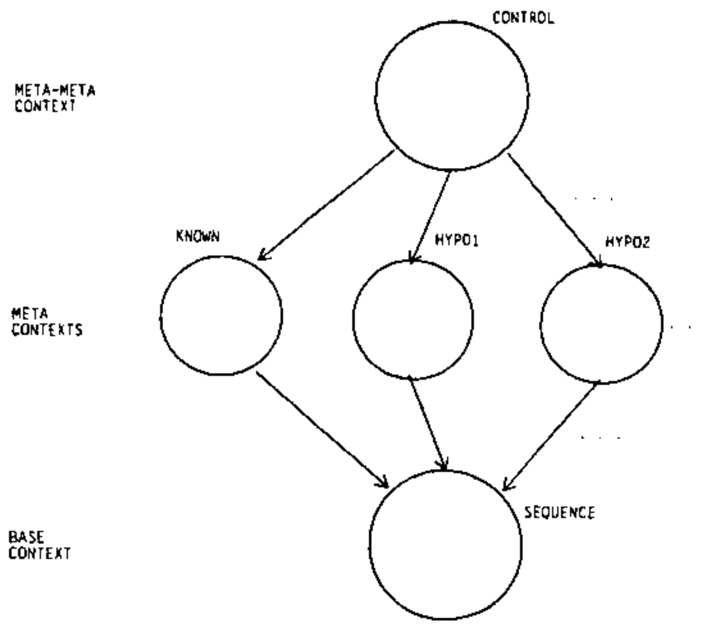

Figure 2. The context structure of Mastermind.

The control meta-meta-context has axioms specifying the above strategies and heuristics. Heuristics embedded in programs are difficult to understand and to change. We did not embed our heuristics in a program. Using meta-theory allowed us to describe heuristics declaratively but use them procedurally. This separation illustrates a unification of the procedural/declarative dichotomy. This idea of declaratively describing heuristics at the meta-level parallels Davis' work on Teiresias

The control meta-meta-theory, running under the SPHERE simplifies actually plays the game. The system is intended to incorporate the heuristics of one of the authors. In fact, being less prone to mistakes, it plays slightly better than he.

v VIEWS

One use of contexts is for abstraction: structuring several views of the same situation. Several contexts can share the same partial model, but talk about it in different ways. A major difficulty in implementing multiple views is maintaining consistency between views and the domain, and among views. Our approach encapsulates the details necessary to display each view in a context. These contexts have axioms about the appearance of domain objects. We use meta-level axioms about the relation between views and domain objects to maintain consistency declaratively.

This quality of expression is particularly useful for representing complex objects with computer graphics. Graphics uses geometric objects as symbols for physical objects or concepts. Explicitly representing this symbolic relationship permits straightforward definition, easy modification, and effective use. We wish to build display systems with knowledge about the domain, the form in which it appears, and the mapping between this form and its content. Contexts facilitate keeping these various functions separate and in an explicit form. This section describes a system with several display forms and an explicit mapping between display forms and their content.

Rubik's Cube [12] is both physically and mathematically a complex object. We can independently represent and display it from either of these views, treating it as a problem of turning sides and matching colors or as a problem in group theory. Different viewB require different display notations. All are helpful in coming to understand the Cube.

We built a seven-context system for manipulating the Cube. We encapsulated each of three display views in a separate context. Two of these show the physical appearance of the Cube. They are 3-D projections of all six cube faces that differ in the orientation of the "hidden" back faces. The third is a table of colored squares representing the position of cube lets in time as the Cube is turned: rows representing successive moves, and columns, positions. This "wallpaper-like" display highlights the periodicity of sequences of moves. The cubelets that move in the cycle contrast with those that stay still. Figure 3 shows the contexts of the Cube system.

Each display context has its own ontology. The 3-D display contexts have terms for subfaces, colors, and cubelets; functions that relate subfaces to colors; and functions that relate sets of subfaces to cubelets. In the table context, color corresponds to the identity of cubelets, not the visible colors of the Cube. Shades of colors correspond to twists and flips [12]. For example, the constant *red* is declared in both the 3-D display contexts and the table context, but has a different meaning in each. Context separation keeps the programmer from confusing these different meanings.

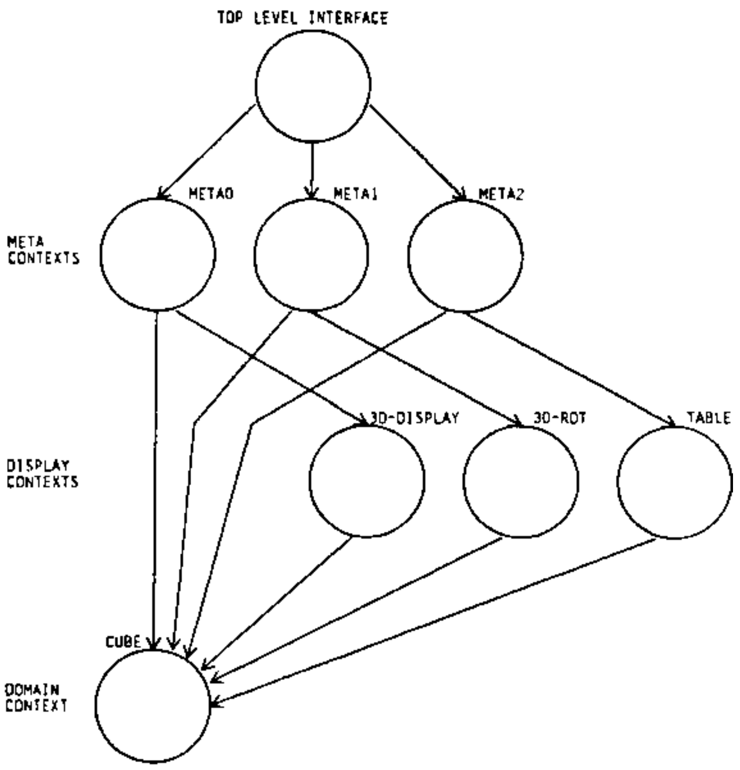

Figure 3. The context structure of the cube system.

A fourth context stores the state of the Cube. Each of the three display contexts is attached to the Cube context both directly, for display purposes, and indirectly, through its own meta-context. They maintain the display relationship between graphic symbol and object. These meta-contexts share a language for manipulating the Cube and for displaying the results of manipulation. Constants of type *operator* in the meta-contexts are attached to subordinate-context functions to move the Cube. Typical primitive operators are RIGHT (turn the right face 90° clockwise) and FRONT-1 (front inverse; turn the front face 00° counterclockwise). More complicated operators are built using functions for concatenation, repetition and taking the inverse. Operators can be simplified syntactically. For example, fourfold repetition of any primitive operator simplifies to the identity operator. We have similar axioms for other simplifications. Thus, much can be simplified symbolically without actually "manipulating" the Cube.

Operators are constants and by themselves do not change the Cube. To actually change a *cube,* we apply the function *MOVE* to an operator and the Cube. Axioms that simplify MOVE-terms both change the CUBE context and invoke the current display context to show the results. This structure maintains consistency between domain objects and the corresponding display objects.

With this system we can apply complex manipulations to the Cube in a language that can be syntactically simplified. We can view the results of this manipulation in one format and then switch contexts to view the Cube in another. This ability to associate linguistic manipulation of operators with different graphic displays facilitates mastery of the Cube.

VI CONCLUSION

In this paper we summarized three examples illustrating the power and flexibility of representation using meta-theory. Each of these issues (modality, heuristics, and views) is difficult to express in conventional AI terms. With the appropriate quality of meta-theoretic machinery, we have uniformly expressed and manipulated them all with a single representation system. Three ingredients contributed to the success of these representations: encapsulation, the clean separation of language and met a-language, and the ability to recurse on meta-languages. We believe that these ingredients form part of a crucial foundation for building intelligent systems.

Acknowledgments

We thank Jonathan King, Egon Loebner, Anne Paulson, Bert Raphael and Richard Weyhrauch for their helpful comments on this paper. We also thank Steve Gadol, Bob Kanefsky, Anne Paulson and Richard Weyhrauch for their participation in building the SPHERE system.

References

[1] Appelt, D. E., "A planner for reasoning about knowledge and action," *Proceedings AAAI-80*, Stanford (1980), pp. 131-133.

[2] Barr, A., "Meta-knowledge and cognition," *Proceedings IJCAI-79*, Tokyo (1979), pp. 31-33.

[3] Bobrow, D., and T. Winograd, "An overview of KRL, a knowledge representation language," *Cognitive Science I*, 1 (1977), pp. 1-46.

[4] Bowen, K. A., and R. A. Kowalski, "Amalgamating language and metalanguage in logic programming," in K. L. Clark, and S.-A.

Tarnlund (Eds.), *Logic Programming,* Academic Press, New York (1982) pp. 153-172.

[5] Conn, A., "High level proof in LCF," *Proceedings 4th Workshop on* Automated Deduction, Austin (1979), pp. 73-80.

[0] Davis, R., "Applications of meta level knowledge to the construction, maintenance and use of large knowledge bases," Memo-283, Artificial Intelligence Laboratory, Stanford University, Stanford (1976).

[7] Gawron, J. M., J. King, J. Lamping, E. Loebner, E. A. Paulson, G. K. Pullum, I. A. Sag and T. Wasow, "The GPSG linguistics system," Proceedings 20th Annua/ Meeting *of the ACL,* Toronto (1982), pp 74-81.

[8] Genesereth, M., "Metaphors and models," *Proceedings AAAI-80,* Stanford (1980), pp. 208-211.

[9] Genesereth, M., R. Greiner, and D. Smith, "MRS manual," Memo HPP-80-24, Heuristic Programming Project, Stanford University, Stanford (1980).

[10] Goldstein, I. P., and R. B. Roberts, "Using frames and scheduling," in P. H. Winston, and R. H. Brown (Eds.), *Artificial Intelligence: An MIT* Perspective, vol. 1, MIT Press, Cambridge MA (1979), pp. 251-284.

[11] Hewitt, C , G. Attardi, and M. Simi, "Knowledge embedding in the description language Omega," *Proceedings AAAI-80.,* Stanford (1980), pp. 157-164.

[12] Hofstadter, D. R., "Metamagical themas," Scientific *American 244,* 3 (1981), pp. 20-39.

[13] Knuth, D., "The computer as Mastermind," *Journal of Recreational Mathematics 9,* 1 (1976), pp. 1-6.

[14] Konolige, K., "A metalanguage representation of relational databases for deductive question-answering system," *Proceedings IJCAI-81,* Vancouver (1981), pp. 496-503.

[15] Konolige, K., and N. J. Nilsson, "Multiple-agent planning systems," *Proceedings AAAI-80,* Stanford (1980), pp. 138-142.

[16] Kornfeld, W. A., and C. E. Hewitt, "The scientific community metaphor," *IEEE Transactions on Systems, Man, and Cybernetics 11,* 1 (1981), 24-33.

[17] McDermott, D. V., and G. J. Sussman, "The Conniver reference manual," Memo 259, Artificial Intelligence Laboratory, MIT, Cambridge, MA (1972).

[18] Rulifson, J. F., J. A. Derksen, and R. J. Waldinger, "QA4: a procedural calculus for intuitive reasoning," AI-Technical Note 73, SRI, Menlo Park, CA (1972).

[19] Smith, B. C, "Reflection and semantics in a procedural language." TR-272, Laboratory for Computer Science, MIT, Cambridge, MA (1982).

[20] Weyhrauch, R. W., "Prolegomena to a theory of mechanised formal reasoning," Artificiai *Intelligence 13,* 1 (1980), pp. 133-170.