A.I. Memo No. 681   July 1982

SUPPORTING ORGANIZATIONAL PROBLEM SOLVING

WITH A WORKSTATION

Gerald Barber

Abstract

This paper describes an approach to supporting work in the office. Using and extending ideas from the field of Artificial Intelligence (AI) we describe office work as a problem solving activity. A knowledge embedding language called Omega is used to embed knowledge of the organization into an office worker's workstation in order to support the office worker in his or her problem solving. A particular approach to reasoning about change and contradiction is discussed. This approach uses Omega's viewpoint mechanism.

Omega's viewpoint mechanism is a general contradiction handling facility. Unlike other Knowledge Representation systems, when a contradiction is reached the reasons for the contradiction can be analyzed by the deduction mechanism without having to resort to a backtracking mechanism.

The Viewpoint mechanism is the heart of the Problem Solving Support Paradigm. This paaradigm supplements the classical AI view of problem solving. Office workers are supported using the Problem Solving Support Paradigm.

An example is presented where Omega's facilities are used to support an office worker's problem solving activities. The example illustrates the use of viewpoints and of Omega's capabilities to reason about it's own reasoning process.

1. Introduction

This paper describes an approach to supporting work in the office. Using and extending ideas from the field of Artificial Intelligence (AI) we describe office work as a problem solving activity. A knowledge embedding language called Omega is used to embed knowledge of the organization into an office worker's workstation in order to support the office worker in his or her problem solving. Omega's Viewpoint mechanism is used to reason about change and contradiction.

In the following section we introduce our abstract characterization of organizations under the name of Office Semantics. Following this we discuss the character of the knowledge used in organizational work and the problem solving characteristics of office work. In the next section we consider the problem of describing office work and how this is best done in terms of the goals of the organization and the actions of its members. The next section concerns the AI problem solving paradigms applied to office work. We discuss how the classical AI view of problem solving is not appropriate and propose to supplement it with the Problem Solving Support Paradigm. The next two sections concern Viewpoints and contradiction handling in Omega.

2. Office Semantics

Office Semantics is the study of information intensive organizational work. Its name reflects the concern with *the intent behind the act*. Office Semantics is concerned with understanding the reasons behind the physical tasks that are performed in organizational work. To understand organizational behavior a distinction is made between the *application structure* of the organization and its *organizational structure*, as illustrated in the diagram below.
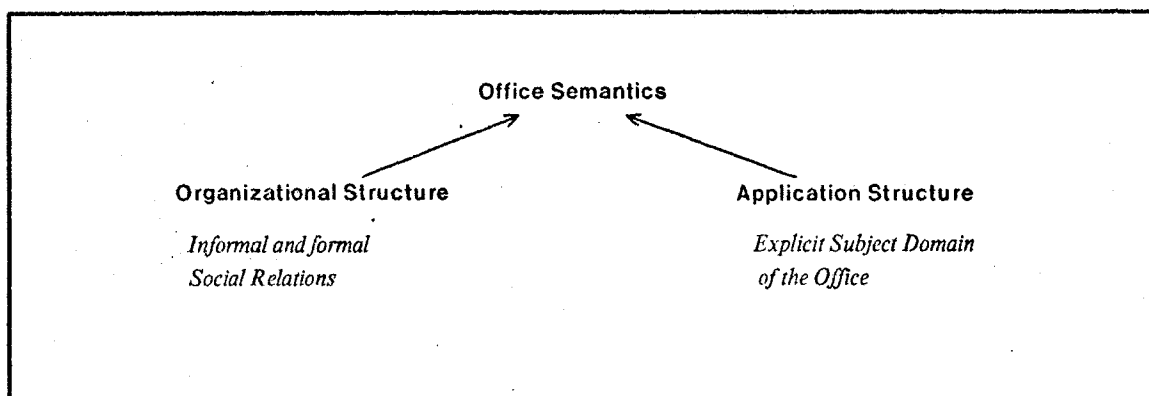
Figure 2-1: Office Semantics: Application and Organizational Domains

We make the distinction between the application and organizational structures because they are bodies of knowledge that often react to different forces of change and because of their differing functions in the

organization. The organizational structure changes in response to forces such as work force mobility and change in the formal structure of the organization. The application structure responds to changes in laws governing aspects of the application, for example tax laws. Changes in product and service requirements also change the application structure. The organizational structure realizes the problem solving strategies necessary to fulfill the requirements of the application structure. For this reason the application and organizational structures are inter-dependent in the task of achieving the organization's goals. The distinction between the application and organization structures does not imply that one is more relevant to organizational work than the other. Organizational work must conform to the constraints and rules derived from both the organizational and application structures.

3. Office Work as Problem Solving

A fundamental premise is that problem solving is basic to office work. Office work has four fundamental characteristics shown below.
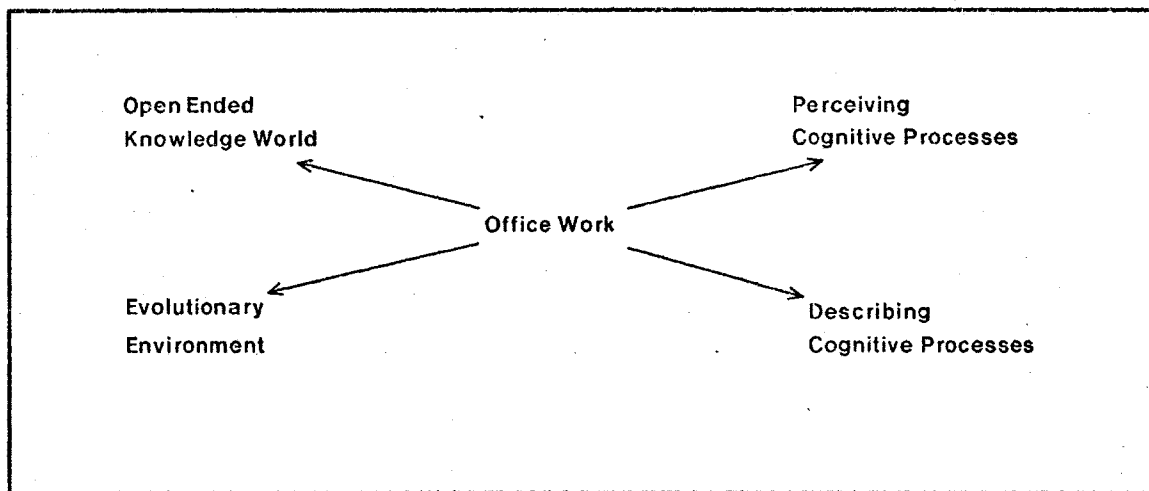
Figure 3-1: Characteristics of Office Work

3.1. Open-Ended Knowledge World

In contrast to some knowledge domains investigated by AI researches such as the Blocks World [Winograd 71] the world of organizational knowledge is not closed. Fikes, Henderson, and Suchman at the XEROX Palo Alto Research Center have described in detail the open-ended characteristic of the organizational world in [Fikes, Henderson 80, Suchman 79]. The complete set of actions relevant to the organizational world is unknown and unknowable. The set of all possible states are unknowable as are all possible alternatives for

achieving a goal. The result is that unforeseen situations are a common occurrence. This is as much a property of the perceiver of the world as it is of the world itself since it is our assumption that the perceiver is of limited cognitive capabilities.

The open-ended character of the organizational knowledge world places demands on the kind of description system used to describe organizational knowledge. In particular the description system must be able to assimilate new information about actions, situations, and alternatives to achieving goals in an incremental fashion. The description system must be able to reason with partial information about problem solving states.

3.2. An Evolutionary Environment

Organizational are continuously changing. Any attempt to understand and describe organizational behavior must cope with the problem of trying to describe a dynamic, evolving system. This is a central problem both in talking about organizations and in doing work within organizations. A description system must be able to describe an organization that is continuously changing. A description system must also furnish tools to manage change so office workers may use it in performing their tasks. The Omega description system provides the Viewpoint mechanism to describe and reason about change.

3.3. Perception of Cognitive Processes from Overt Physical Actions

Trying to understand what task someone is doing and the reasons for each action performed in carrying out a task by watching the person perform a task is in general not possible. Information used in performing the task is not manifest in the physical actions the task entails. Additionally, introspection gives at best partial, and often apparently contradictory information. This characteristic implies two limitations on office knowledge: first, the quality of information gathered by observation or interview is limited. Second, and hence, more effective methodologies are desirable. Partly because of this problem the approach we take is to *support* problem solving rather than replace the individuals in an organization doing the problem solving.

3.4. Describing Cognitive Processes

In order to discuss cognitive processes they must be describable. In the general case it is not possible to describe all mental activity involved when a person is thinking through a problem. Describing all mental activity involved is also not desirable since much of it may be irrelevant to the problem at hand and idiosyncratic to the particular individual. The goal of describing cognitive processes is not to develop a psychological theory of the individual in an organizational setting but to describe the individual's task relevant knowledge and thought processes in a way that--taken in aggregate--explains organizational behavior.

The premise is that there is a way to describe an *organizational person* in terms of application and

organizational knowledge. In adopting this premise an assumption is made that an organization works in such a way as to factor out the individual idiosyncrasies of its members. This can be seen by the fact that as organizations grow in size their members have less of an impact on the organization's characteristics. The reason for making this assumption is that many organizations have similar behaviors but are made up of diverse personalities.

4. Describing Office Work

The purpose of describing office work is not only to make a record of the activities involved in the performance of office tasks but to uncover the implicit assumptions of the work. The work description includes the mental and physical activities that an office worker engages in and the reasons for these activities. An approach to characterizing work in the office is to consider it as organized in procedures in a fashion similar to the computer science notion of procedure. In this way office work would be described as a sequence of steps with decision points to manage flow of control.

4.1. Pitfalls of a Procedural Description Methodology

A procedural characterization is problematic for several reasons. Even routine tasks in offices are beset by unexpected obstacles. In a procedural approach it is necessary to foresee the possible alternative courses of action when a procedural step cannot be performed. Determining what the alternatives are is part of what office work is; all alternatives cannot be determined in advance. As a result a procedural approach is not a very useful style of work description because it needs to be augmented by the procedure's goal structure. When a procedure augmented in this way then one can examine the procedure's goal structure in order to generated alternative steps when a step cannot not be performed. Interesting studies along these lines are contained in [Suchman 79, Fikes, Henderson 80].

4.2. Explicit Representation of Goals and Actions

A description of office work in terms of goals and actions is a direct way of characterizing office work. A procedural description of an order entry task, for example, succinctly characterizes the important points of the task. But precisely because of its succinctness a procedural description suffers from two defects: first, it glosses over minor details that may be problematic or critical in practice; second, the reasons for the actions specified by a procedural description must be inferred. Thus if it is impossible to fulfill a requirement in the procedural description, such as obtain the delivery address for an order, the office worker must rely on intuition and experience to select an alternative action. The more desirable approach is to state explicitly the reasons the action is needed--the goals the action achieves.

The explicit representation of goals and actions provides a recourse to handle unexpected contingencies.

Office workers are able to handle unexpected contingencies in their daily work because they know the goals of the office work and because they know what actions are needed to achieve the goals of the office work. These goals and actions are often implicit in the work and in the office workers' knowledge of their work. If a particular action cannot be performed the computer system can possibly suggest an alternative action. Failing this the office worker can use the computer system to examine the goals an alternative action must inherit from the action that cannot be performed. Together, the office worker and computer system can construct a new plan of action that maintains the necessary constraints and makes progress toward achieving the goals in question.

To support the problem solving activity in office work knowledge about the goals and constraints of the office work are explicitly represented. This builds a teleological structure of the office work within the computer. Actions that would be performed during the course of the office work are linked to the reasons they are performed and to the constraints that they are required to maintain. Explicit representation of the goals and actions exposes hidden assumptions and implicit goals about the office work. In addition, explicit representation makes the actions performed by an office worker more understandable by machine or by another individual.

Added coherence between different functional elements of a system has the benefit that the user's actions and the goals of the office procedure can be understood in terms of each other. It is useful for the system to understand the goals in order to interpret the user's requests and suggest problem solving tools for achieving the goals. In turn the user's actions suggest what the current goals are and narrows the variety of problem solving methods and size of the solution space. Discussion about characterizing office work in terms of goals is contained in [Fikes 80, Barber 82].

5. Problem Solving Paradigms

A classical problem solving paradigm in AI is depicted in the diagram below. However, this paradigm is difficult to apply in the organizational world. The organizational world differs from the traditional AI worlds such as cryptarithmetic or the blocks world in that: it is distributed and parallel thus there is more than one individual working on the problem; and it is open ended in the sense that all actions and consequences of actions are not known.
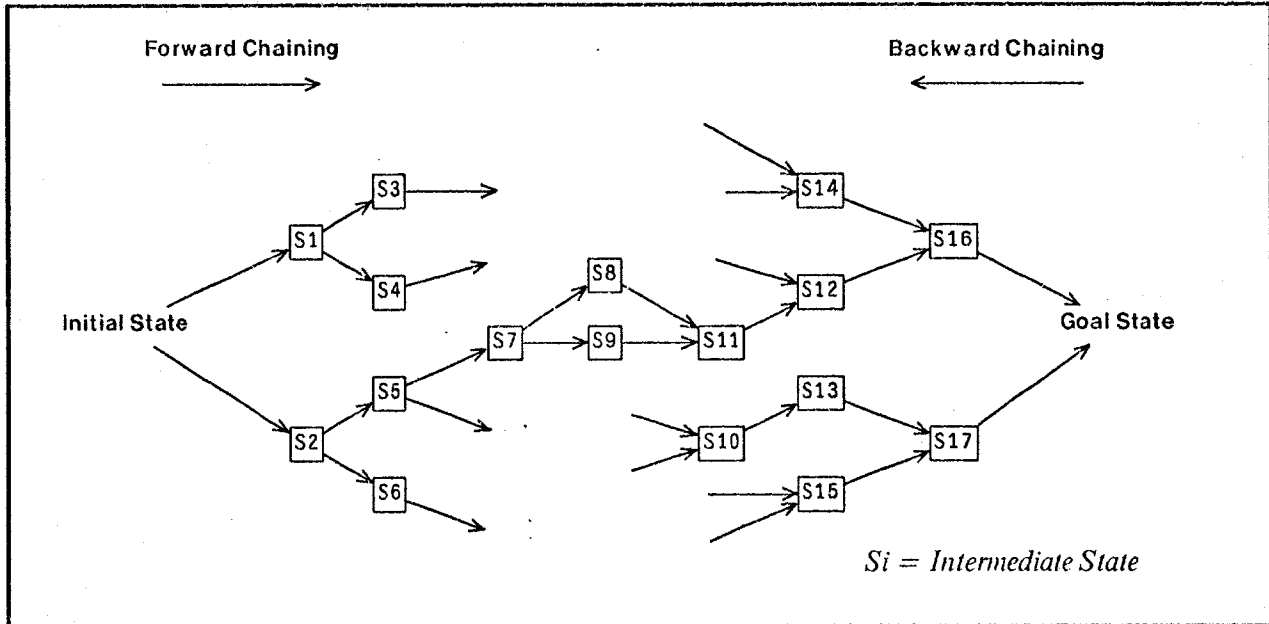
Figure 5-1: The Classical AI Problem Solving Paradigm

In the above view of problem solving the problem solver is given a well defined initial state, for example the configuration of a chess board, a well defined final state, to win the game, and a finite collection of actions or state transformers. The characterization of problem solving is as a search for the sequence of actions that will achieve the goal state. The test to see if the goal state has been achieved is objective and two valued, either the goal is achieved or not. The problem solver is assumed to be a single individual, for example a single chess player playing an opponent as opposed to a team of chess players playing an opponent. Thus there are no problems with synchronization or conflict with other problem solvers. When more than one problem solver is cooperating on a problem, as in an organization, a global state description of the problem and the problem solver is no longer practical.

This is a seductive paradigm but it is hard to apply in the organizational setting. The reason for this is that in using this paradigm one determines a possible means to achieve a goal by examination of the current and goal states. But in many cases in office work the goal is vague and how much information is relevant to achieving the goal is not clear; this makes an assessment of the current state difficult. This problem is suggested by the case studies in [Wynn 79] and has been pointed out by [Suchman 79].

In addition the purpose of this paradigm is to do the problem solving. Our approach is to support problem solving. As a result we extend the above view of problem solving with the *Problem Solving Support Paradigm* shown below.
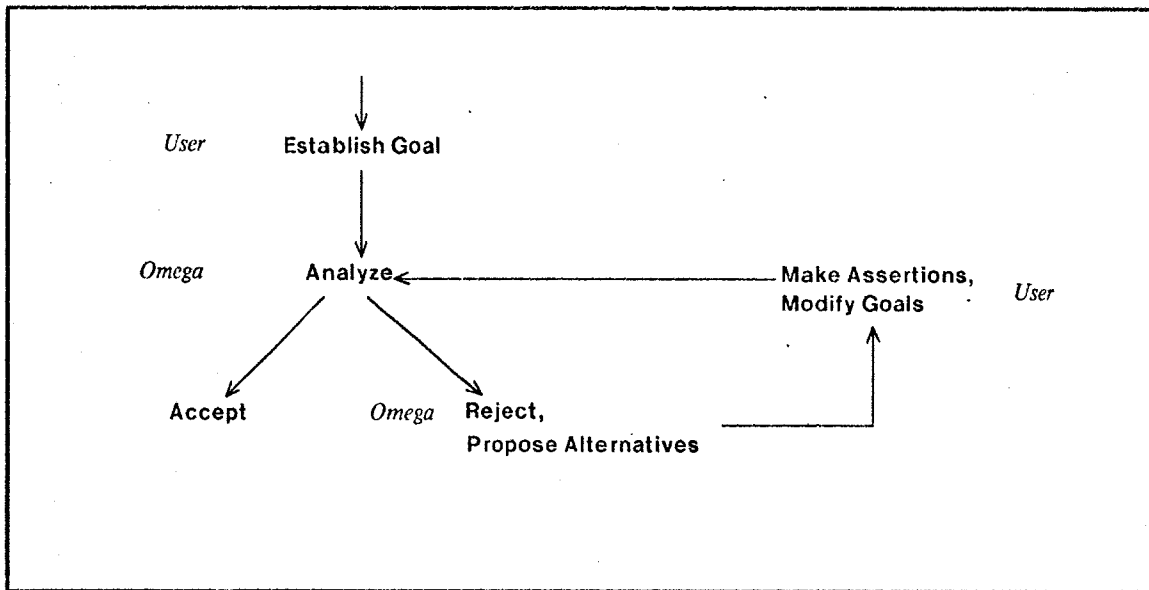
Figure 5-2: The Problem Solving Support Paradigm

In the problem solving support paradigm the office worker establishes a goal, for example to send a message or to complete a step in an office procedure. Based on what Omega knows about the goal, Omega either tries to establish the goal or to refute the goal. If is is not possible for Omega to establish the goal Omega notifies the office worker that the goal cannot be established or that contradictory information has been discovered during the attempt to establish the goal. At this point the office worker can either modify the goal or make further assertions possibly supplying necessary information to establish the goal. Omega then attempts to establish the goal again. This cycle continues until the goal is established. The analysis is accomplished using Omega's viewpoint mechanism.

6. Supporting Office Work

Omega provides a uniform framework within which to implement tools to support an office worker's problem solving. This has the benefit that different tools may cooperate easily in achieving the goals of particular office tasks.

Knowledge is embedded in the form of descriptions about objects in the system and the relationships between these objects. Office Talk, OBE, and other systems [Ellis, Nutt 80, Zloof 80, Tsichritzis 79] have adopted forms as the basic element of the system, an attempt is then made to represent everything in the system using forms. In contrast to the forms model we adopt a model in which descriptions are the basic element of the system. Since the knowledgebase is represented using Omega's description lattice data does not have to be

cast in a rigid form as it does in traditional data processing applications. The consequence is that office tasks may be reasoned about more on an individual basis.

Among some of the functions that a forms model provides are:

- **Storage of information** as in records.

- **Transfer of information** as in messages.

- **Display of information** in an abstracted and structured manner.

- **Accumulation and modification of information** as the form is used by individuals in the accomplishment of their tasks.

However, descriptions provide much greater functionality than a forms model. Descriptions are a very general facility; not only do they provide the functions that forms-based systems have as shown above but they also are the basis for Omega's reasoning machinery. Descriptions provide:

- **A means for error checking** of information in an office system.

- **A basis for retrieval and deduction** based on stored information.

- A means by which the **structure of office activity and problem solving processes** are described.

- A means by which the **structure of the application and organizational domains** of an organization are specified.

- **Viewpoints** by which change and inconsistent states may be reasoned about.

A central part of Omega's reasoning capabilities is realized using the Viewpoint mechanism. This is described below followed by an example of the use of Omega in supporting an office worker.

7. Viewpoints

Viewpoints may be thought of as repositories for descriptions and thus statements. Viewpoints are reminiscent of McCarthy's situational calculus [McCarthy, Hayes 69] and the contexts of QA4 [Rulifson, Derksen, Waldinger 72] and Conniver [Sussman 72]. The most notable difference between viewpoints and these systems is that viewpoints are objects within the system, they may be reasoned about and described just as any other description in the system. Viewpoints are not restricted to being organized into a tree structure as are the contexts of Conniver and QA4.

A key property of viewpoints is that information is only added to them and is never changed. Consider, for example, a description of an invoice. The description is in a viewpoint and may be further described in the viewpoint increasing its specificity. There may be rules that maintain constraints between attributes of

descriptions, thus as information is added to a viewpoint further information may be deduced. For example, a rule for invoice descriptions may state that the subtotal plus a sales tax must equal the total; thus when any two of the attributes is known the third may be calculated. Should a description in an attribute be changed in a particular viewpoint, for example the subtotal change from \$5 to \$10, then the following scenario might occur:

1. A new viewpoint is created and described as being a successor to the old viewpoint.

2. All descriptions that were not derived from the changed description are inherited to the new viewpoint.

3. The new description is added in the new viewpoint, any deductions resulting from this new information are made.

4. The descriptions in the new viewpoint describe the changed state of the invoice.

In this case the new viewpoint inherits all but the changed description and the descriptions deduced from the changed description from the old viewpoint. What actions are taken when information in a viewpoint is changed is controlled via sprites [Kornfeld 82]. Sprites are procedures that fire when a condition they are watching for arises in the knowledge base. Sprites typically fire when assertions are made or goals are posted. In the example above a simple action is specified: all information not derived from the changed information is inherited into the new viewpoint. Other actions would be to disallow change, in the case of protected information, or to signal a contradiction and allow the user to help resolve it.

7.1. Handling Change

Many approaches have been developed to manage change; we begin with the most simple and proceed to the more sophisticated. In some cases the approach to keeping track of changing information has been via updates to data structures. Systems based on property lists or records such as in Lisp or Pascal have used *put* and *get* types of operations to update and read database information. These are low level operations and have the disadvantage that they provide no support for propagating changes. Thus, deductions based on updated information must be handled explicitly leading to excessive complexity and modularity problems. Languages like FRL [Goldstein, Roberts 77] solve this problem by using triggers on data structure slots (actually the slots of frames), to help propagate changes. The problem with this approach is that there is little support for keeping track of what was deduced and why.

The language KRL has been used to implement a knowledge-based personal assistant called ODYSSEY [Fikes 80]. ODYSSEY aids a user planning travel itineraries by keeping track of what cities a traveler will visit, how the traveler will get to the cities and where the traveler will stay in the cities. In this system *pushers* and *pullers* are used to propagate deductions as a result of updates and to make deductions on reads. A

simple dependency mechanism is used to record information dependencies. The problem is that there is a transition period from the time when a value is changed to the time that all changes are propagated. During this transition period the database is in an inconsistent state and rules may fire making deductions based on inconsistent information. In both KRL and FRL it is necessary to be very careful about the order in which triggers fire for as updates are made there is both new and old information in the database making it difficult to prevent anomalous results due to inconsistencies.

The AMORD system attempts to maintain a globally consistent database at all times. A Truth Maintenance System [Doyle 77] maintains the status of facts, when a fact becomes *outed*, or disbelieved, the status of all facts that depend on the original fact are also set to out. During the period of time that facts are being outed or reinstated the database is unavailable for the firing of rules. Thus when the rules do fire, they always see a consistent database. This system reduces the possibility of making erroneous conclusions considerably at the expense of a global notion of truth and efficiency.

Steele has developed a constraint based programming language [Steele 80]. In this system a network of nodes and connections is used to build a constraint network. Values deduced by rules at the nodes propagate through the network creating a flow of information through the net from input and constant values to deduced values. Like AMORD, Steele's system enforces a global notion of truth. Like KRL, rules can fire on an inconsistent database signaling contradictions. These rules represent false alarms because once the propagation caused by the original change is done, the database is consistent and the fired rules are no longer relevant. Partially because of the "false alarm" problem Steele has devised a system of prioritized queues that defers the processing of fired rules that are likely due to false alarms until rules that are likely to bring the database into a consistent state have fired. Because of the global truth requirement false alarms will not cause inconsistent results as is the case with the KRL system, they will just lead to inefficiency.

A characteristic shared by all these systems is that they are non-monotonic. Information is lost when, for example, values of slots are changed in FRL or when inputs are changed in Steele's constraint system. This is a fundamental limitation because it means that the systems are constrained in their history keeping capabilities. If a value of some parameter in one of these systems is changed from A to B causing the implications of B to be deduced and then it is changed back to A there is no way for the system to know the parameter's value was A at a previous time. One might object that this is not the case in AMORD, that when the parameter is changed back to A the status of the relevant facts are simply changed from *out* to *in* and no recomputation is necessary. This may be true but this behavior is implemented by a mechanism beyond the reach of the system's deduction machinery. There is no way for AMORD to reason about the fact that the parameter's value was A.

8. Contradiction Handling with Viewpoints

The systems described above cannot reason directly about contradictions because they are based on logics where truth is global characteristic of a statement. Since these systems enforce a global notion of truth, when a contradiction exists anything can be derived by their inference rules. Thus when a contradiction is detected the systems deduction machinery is useless. The approach taken by AMORD and Steele's constraint system for example, is to use a mechanism outside the logic to reestablish consistency. Once the world is consistent, the deduction mechanism can operate normally. The Viewpoint mechanism provides a method to quarantine inconsistency to within a viewpoint so that reasoning can be done outside of the inconsistent viewpoint and thus valid conclusions can still be made.

The ability to limit the effect of contradictions to within viewpoints is done by explicitly keeping track of what is believed to be true, i.e. *assertions*, and why it is believed to be true, *justifications*. This information is expressed in the Omega language so it is within reach of the deduction mechanism. It has been stated that a general purpose programming language isn't one if an implementation for the language cannot be written in the language itself [Steele 80]. We agree and recast this statement: a knowledge embedding language isn't one if it can't represent and reason about why it believes what it does believe. Given any statement, Omega can answer whether the statement is believed to be true and why, whether the statement is believed to be false and why, or whether Omega doesn't know.
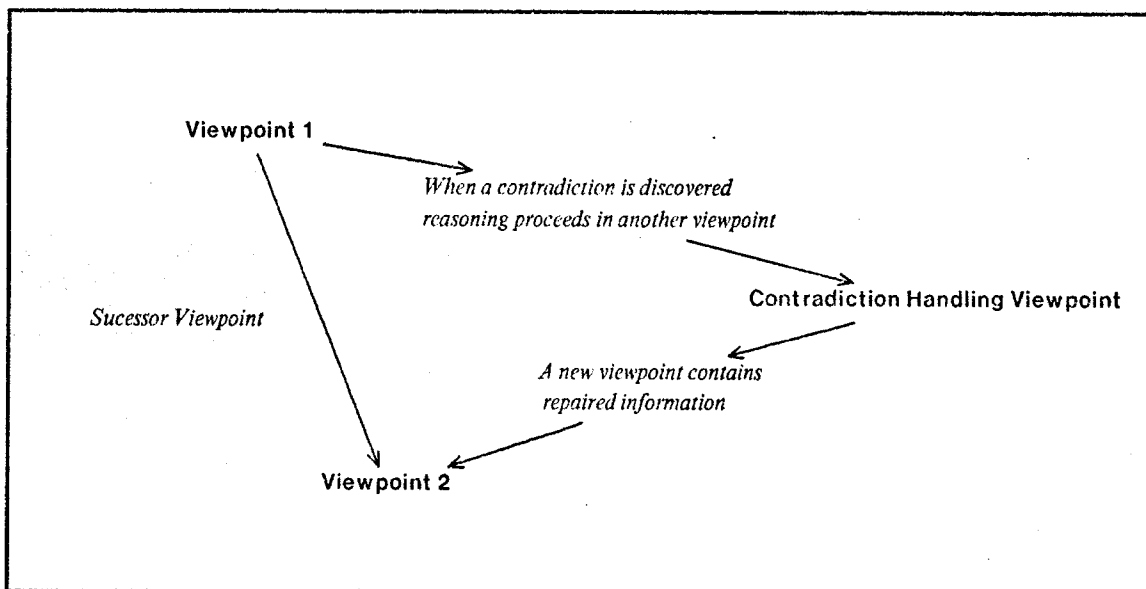

Figure 8-1: Handling Contradictions with Viewpoints

Contradiction Handling in using viewpoints proceeds as indicated in the above diagram. In this example

reasoning in viewpoint 1 has discovered a contradiction, This causes activity to cease in that viewpoint and to switch to the contradiction handling viewpoint. In the contradiction handling viewpoint the justifications for the assertions that are in contradiction are analyzed. The assertions are filtered, depending on the source of the contradiction, and moved to the viewpoint 2. The relationship between the two viewpoints are described and reasoning proceeds in viewpoint 2 where it left off in viewpoint 1.

9. An Example

This section consists of an example of the ideas that have been developed in earlier sections. The main ideas we will address will be the following:

- **Problem Solving Support** - Use of the problem solving support paradigm in helping office workers in their tasks.

- **Goals** - The use of goals to describe office work. How these goal descriptions can help office workers in the performance of their tasks.

- **Contradiction Handling** - examples of Omega's contradiction handling capabilities in dealing with real work knowledge.

The example is taken from an office in the Defense Department that is part of the Officer Transfer Process. This process is the method by which Navy officers are reassigned to tours of duty or *billets*[1] when their present billet assignment expires. The *Assignment Officer* fills the role depicted in figure 9-1 below.
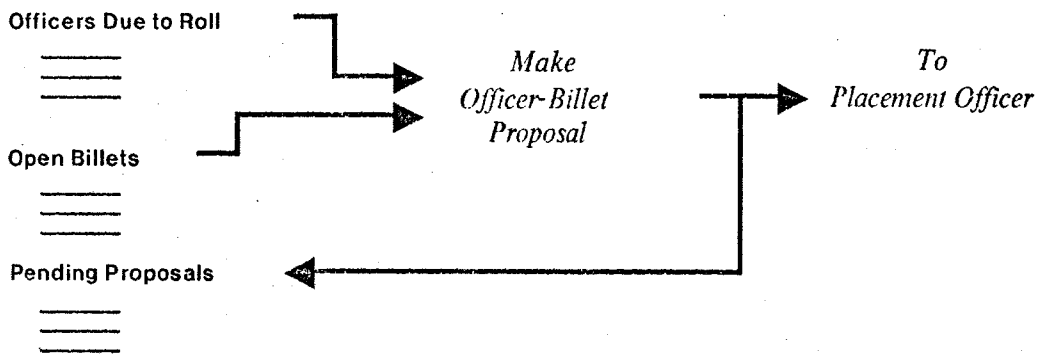

Figure 9-1: The Assignment Officer Role

---

[1]Billets are jobs, an officer is usually assigned to a billet for 3 years.

Conceptually the Assignment Officer's role is simple; he or she has a list of officers that are *due to roll*[2] and a list of open billets. The assignment officer chooses an officer-billet pairing and passes this proposal on to the *Placement Officer* for acceptance and keeping a record of the proposal. The Placement Officer accepts or rejects the proposal. The assignment officer represents the interests of the officers that are due to roll. Thus in each officer-billet proposal the Assignment Officer chooses a billet that will help attain the career objectives of the officer due to roll.

As an example we show how part of the officer transfer process can be described in terms of goals and how this description can help Assignment Officers in their work. The following description focuses on the internal mechanism that Omega uses to reason about a particular domain. We do not describe the user interface with which the user would make assertions or post goals or the way that Omega would present the results of its reasoning processes to the user. Our goal is to first get the underlying mechanism working right and then to work on a user interface for those mechanisms.

9.1. Posting a Goal

Shown below in figure 9-2 is the top level goal for a particular assignment proposal. The goal is to show that OFFICER-6 and BILLET-17 form a reasonable assignment proposal. This goal may have been posted by the assignment officer because he or she wanted to establish that the officer-billet pair formed a reasonable proposal or the goal may be part of a query that is trying to determine all the reasonable proposals for a particular group of officers and billets.

---

[2]An office is due to roll when his or her current billet assignment will expire in 6 months.
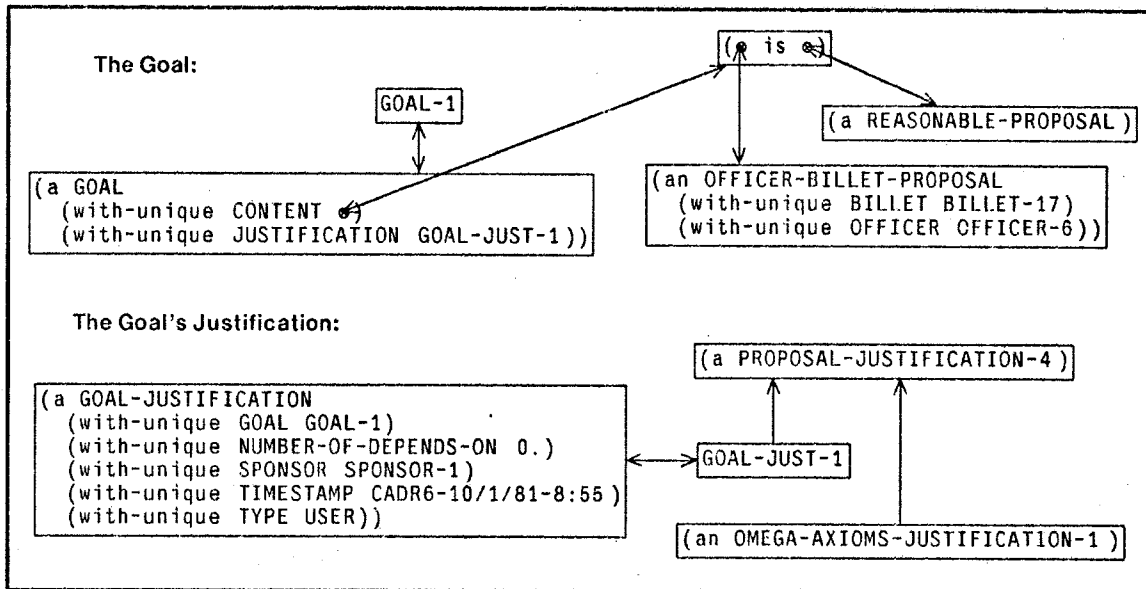
Figure 9-2: The Assignment Proposal Goal

The goal is represented within Omega's description structure. The boxes in the diagram represent descriptions. A single headed arrow indicates that the description at the tail of the arrow is related to the description at the head of the arrow by the inheritance or *is* relation. A double headed arrow indicates two descriptions that are *same*, or inherit from each other. Arrows that point inside a box indicate components of the description represented by the box. Viewpoints are collections of goals and assertions. In this diagram **Goal-Just-1** is depicted as being in viewpoint **Proposal-Justification-4** by the inheritance relation.

As can be seen above the goal has two parts, a content and a justification. The content is the logical statement of the goal and the justification is the reason the goal was posted. In this case the goal is that a particular officer-billet proposal is reasonable. Various information is registered in the justification including: the goal type, **User** which means that this is a top level goal entered by the user; the goal timestamp, when and where the goal was posted; and the sponsor. The sponsor allocates computational resources toward establishing the goal. A sponsor is given a certain quanta of resources with which to establish a goal. If the sponsor runs out of quanta before establishing the goal it must ask for more quanta before it can proceed. If the goal is achieved or if it is shown to be unachievable then the sponsor may be *stifled*. When a sponsor is stifled no further processing can proceed to establish the goal under the auspices of the sponsor. The use of sponsors in Omega is based on the work of Kornfeld as described in [Kornfeld 79].

Now suppose that Omega has been told the following about what constitutes a reasonable proposal:

```
(⇒ (∧ ≡B is (a Billet-Fulfilling-Career-Objectives
              (with unique Officer ≡O))
     ≡O is (a Qualified-Officer
              (with unique Billet ≡B)))
  (is (an Officer-Billet-Proposal
          (with unique Billet ≡B)
          (with unique Officer ≡O))
      (a Reasonable-Proposal)))
```

In the above implication the "≡" symbol is used to mark universally quantified variables. Thus this implication states that an Officer-Billet proposal is reasonable if the officer is a qualified officer for the particular billet and if the billet fits in with the officer's career objectives. If and when Omega decides that a particular assignment is reasonable is only according to the definition Omega has concerning what it takes to be a reasonable proposal. The above goal would be used as a filter to pick out the most obvious characteristics of the proposed assignment. The Assignment officer may look at a proposal that Omega has judged reasonable and reject it because of some criteria that Omega does not know about.

9.2. Posting of Subgoals

The assertion of the above rule creates several sprites,[3] one of which looks for a goal that matches the consequent of the implication. If the sprite fires it posts the antecedent of the implication as a goal. The sprite then creates a second sprite that watches to see if the antecedent is asserted. When the antecedent is asserted the second sprite fires and asserts the consequent. Thus as a result of the above implication and the goal in figure 9-2 the following subgoals will be posted.

---

[3] A sprite watches for the assertion of an implication. When the sprite fires on such an assertion it creates 4 sprites corresponding to the 4 ways the implication can be used. These correspond to the antecedent and consequent reasoning of the implication and its contrapositive.
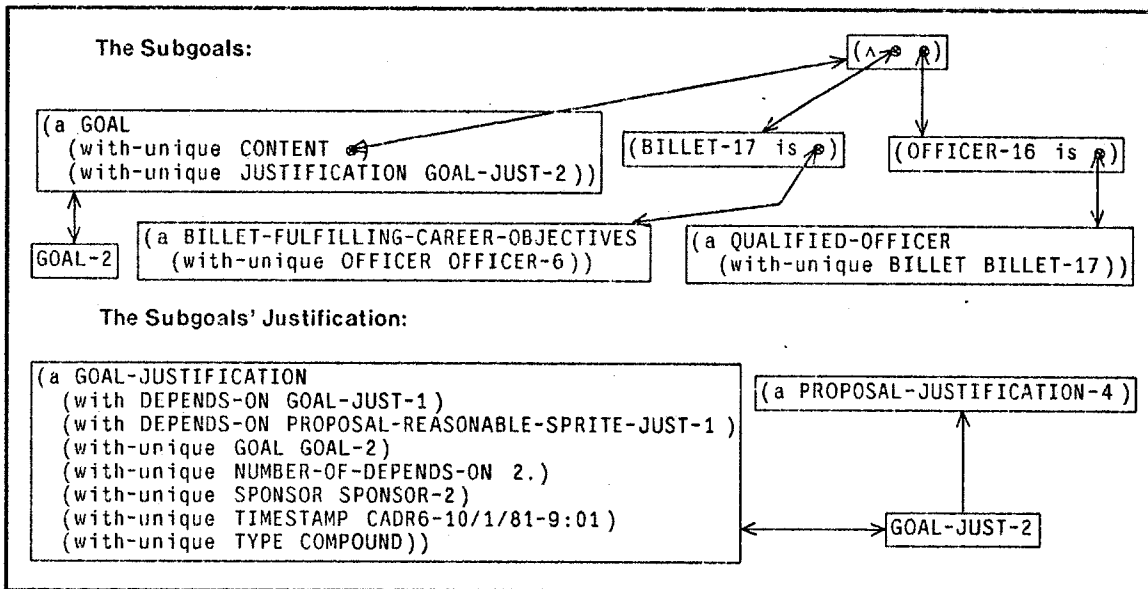
Figure 9-3: The Assignment Proposal Subgoals

Notice that the justification for this subgoal contains a new sponsor SPONSOR-2. The sprite that created the new subgoal also created a new sponsor for the processing that attempts to establish the subgoal. The reason for this is so that when the subgoal is achieved (or shown to be unachievable) the subgoal can be stifled without affecting the processing of the supergoal. In addition the sprite also linked the subgoal to the goal by setting up the following *is* relation:

> GOAL-JUST-1 *is* (*a* Goal-Justification
>   (*with* Depended-on-by GOAL-JUST-2))

This enables analysis of the reasoning when, for example, a goal cannot be achieved or is shown to be unachievable because of the failure of some subgoal. It is also useful when Omega explains how it has achieved a goal.

A conjunctive goal as in the diagram above is handled in the following fashion, a sprite will notice that there is a conjunctive goal. The sprite will fire and post the two conjuncts as goals. In addition the sprite will create additional sprites that watch for the assertion of each of the conjuncts or negation of either conjunct. When both conjuncts are asserted the conjunction is asserted; if either conjunct is negated the negation of the conjunction is asserted.

Suppose the following knowledge is stored in the description lattice with relevance to the goal shown above. Note that for brevity we do not include the assertions and justifications in this diagram, we just illustrate the *is* relations directly. The reader will note that the officer fulfills the billet prerequisites for past billets but not

those for schooling. We describe how Omega discovers this.

```
                        (an OFFICER
                          (with NUMBER-OF-PAST-BILLETS  2.)
                          (with NUMBER-OF-SCHOOLING  2.)
                          (with PAST-BILLET DESK-JOB)
                          (with PAST-BILLET SAILOR)
 OFFICER-6                (with SCHOOLING ADMINISTRATION)
                          (with SCHOOLING LIFE-AT-SEA)
                          (with-unique NAME Juan Diaz)
                          (with-unique ULTIMATE-CAREER-OBJECTIVE  PILOT))


 (a CAREER-OBJECTIVE )    (a BILLET
                            (with PREREQ-BILLET DESK-JOB)
                            (with PREREQ-BILLET SAILOR)
                            (with PREREQ-SCHOOLING GROUND-SCHOOL)
             BILLET-17     (with-unique NUMBER-OF-PREREQ-BILLET  2.)
                           (with-unique NUMBER-OF-PREREQ-SCHOOLING  1.)
 PILOT                     (with-unique TYPE PILOT))
```

Figure 9-4: Some Officer and Billet Knowledge

In this discussion we concern ourselves with how Omega shows that **OFFICER-6** is a qualified officer. The method used to show that **BILLET-17** fulfills the officer's career objectives follows in a similar manner. Omega has been given the following equivalence concerning qualified officers.

$$(\Leftrightarrow (\wedge \equiv O \text{ } is \text{ } (an \text{ Experienced-Officer}$$
$$(with \text{ } unique \text{ Billet } \equiv B))$$
$$\equiv O \text{ } is \text{ } (a \text{ Schooled-Officer}$$
$$(with \text{ } unique \text{ Billet } \equiv B)))$$
$$(is \equiv O \text{ } (a \text{ Qualified-Officer}$$
$$(with \text{ } unique \text{ Billet } \equiv B))))$$

As in the previous implication, when this equivalence is asserted sprites are created that watch for goals that match either the left or right halves of the equivalence. When a sprite fires after matching one half of the equivalence as a goal it posts the other half as a goal. In addition sprites are created that watch for the assertion or negation of either side of the equivalence. Thus when an assertion or negation of one side is made, the assertion or negation of the other side is made. Thus we have the following subgoals posted with a new sponsor:

```
More Subgoals:                                        >(Δ→● ●)
┌──────────────────────────────────────┐
│(a GOAL                               │    ┌──────────────────┐   ┌──────────────────┐
│   (with-unique CONTENT ●<            │    │(OFFICER-6 is ●)  │   │(OFFICER-6 is ●)  │
│   (with-unique JUSTIFICATION GOAL-JUST-3))│ └──────────────────┘   └──────────────────┘
└──────────────────────────────────────┘

┌────────┐  ┌──────────────────────────────┐   ┌──────────────────────────────────┐
│GOAL-3  │  │(a SCHOOLED-OFFICER           │   │(an EXPERIENCED-OFFICER           │
└────────┘  │   (with-unique BILLET BILLET-17))│ │   (with-unique BILLET BILLET-17))│
            └──────────────────────────────┘   └──────────────────────────────────┘

And Their Justifications:

┌─────────────────────────────────────────────────┐   ┌────────────────────────────┐
│(a GOAL-JUSTIFICATION                            │   │(a PROPOSAL-JUSTIFICATION-4)│
│   (with DEPENDS-ON GOAL-JUST-2)                 │   └────────────────────────────┘
│   (with DEPENDS-ON QUALIFIED-OFFICER-SPRITE-JUST-1)│
│   (with-unique GOAL GOAL-3)                     │
│   (with-unique NUMBER-OF-DEPENDS-ON 2.)         │
│   (with-unique SPONSOR SPROSOR-3)               │          ┌──────────┐
│   (with-unique TIMESTAMP CADR18-10/1/81-9:06)   │<────────>│GOAL-JUST-3│
│   (with-unique TYPE COMPOUND))                  │          └──────────┘
└─────────────────────────────────────────────────┘
```
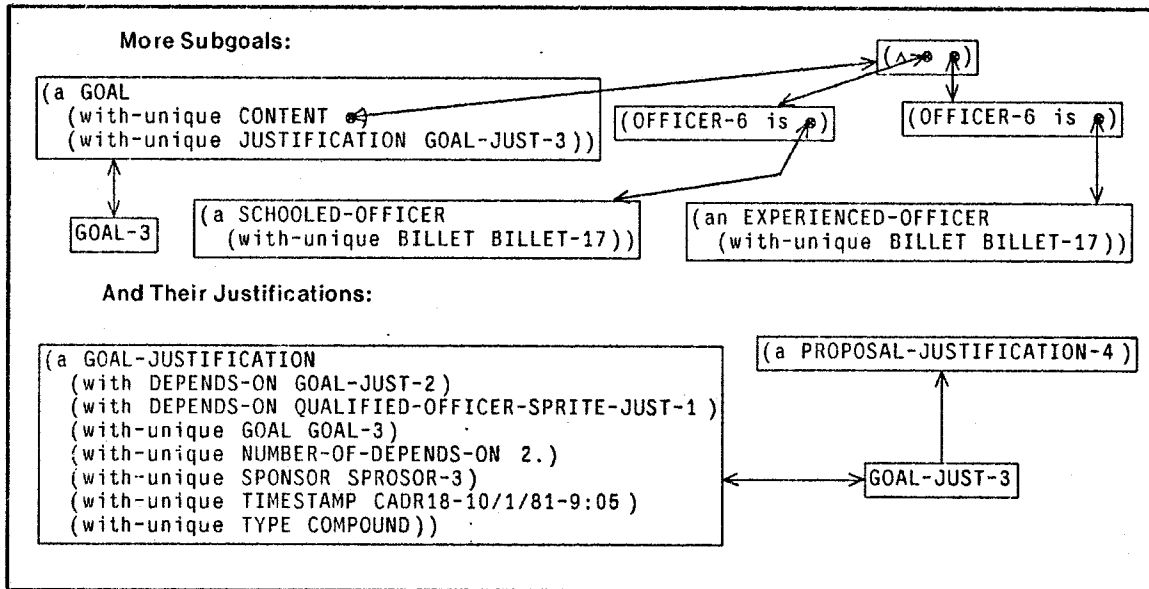
Figure 9-5: Subgoals to Establish Qualified Officer Status

Omega has been told the following concerning what it takes to be a Experienced Officer.

$$(\Leftrightarrow (\textit{for all} \equiv P$$
$$(\Rightarrow \equiv B \textit{ is } (a \textit{ Billet}$$
$$(\textit{with unique} \text{ Prereq-Billet} \equiv P))$$
$$\equiv O \textit{ is } (an \text{ Officer}$$
$$(\textit{with} \text{ Past-Billet} \equiv P))))$$
$$(\textit{is} \equiv O (an \text{ Experienced-Officer}$$
$$(\textit{with unique} \text{ Billet} \equiv B))))$$

This goal is more difficult to achieve. This rule states that if it is true that for every prerequisite of a billet the prerequisite is a past billet for an officer then the officer is an experienced officer for the billet and vice-versa. As with the previous equivalence the right half of the statement will be posted as a goal when the left half is posted as a goal. Since this new goal involves a universal quantification some knowledge of the domain over which the variable ranges is necessary. This is the purpose of the **NUMBER-OF-PREREQ-BILLETS** and the **NUMBER-OF-PAST-BILLETS** attribute descriptions.

9.3. A Subgoal is Established

The method used to prove the universally quantified statement is to first retrieve the number of prereq billets and the number of past billets via sprites. The general approach is to insure that all the prerequisites are past billets; this means we must retrieve all the prerequisite billets. We know when we have retrieved them all by the **NUMBER-OF-PREREQ-BILLETS** number. Once they are all retrieved we check to see that each is a

past billet. If each is a past billet then the universally quantified statement is asserted. If one prerequisite is not a past billet (which we can know since we know how many there are) then the negation of the statement is asserted. If there is not enough information to determine the truth or falsity of the statement the sprites remain waiting for additional information. Once the necessary information is known, if the sponsor of the sprites is still active, the statement or its negation will be asserted.

The reason that the NUMBER-OF-PAST-BILLETS attribute is necessary is so that Omega can know when to stop looking for billets. Without the number stated explicitly Omega cannot conclude that an officer has 2 past billets only because that is all the information that is stored explicitly in the description system. For example, it may be possible to prove the existence of more billets than are explicitly known about. Without explicitly stating the number of past billets the question of whether all billets are known or not is undecidable. This is an example of how Omega's goals of monotonicity and assimilation of new information affect Omega's reasoning processes.

In our example the sprites, using the information that appears in figure 9-4, will conclude that the officer is an experienced officer and will make the assertion shown below.
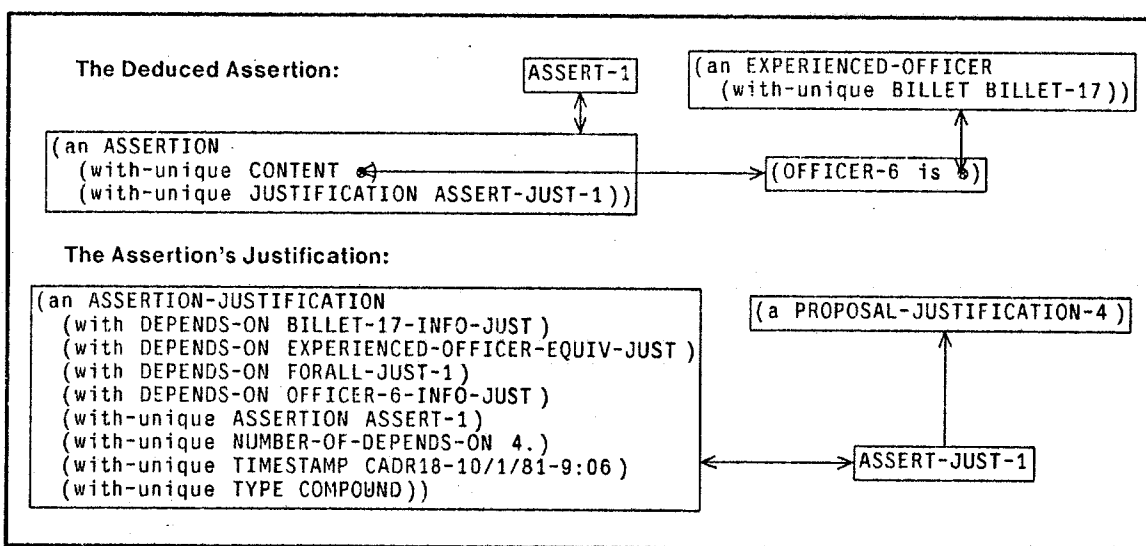

Figure 9-6: The Experienced Officer Assertion

Notice that this assertion depends on the information shown in figure 9-4 BILLET-17-INFO-JUST, OFFICER-6-INFO-JUST, on the justification for the universally quantified statement, FORALL-JUST-1, and on the justification for the equivalence statement EXPERIENCED-OFFICER-EQUIV-JUST. In particular it does not depend on any of the goals that were posted in the process of achieving the goal; as pointed out in [de Kleer, Doyle, Steele, Sussman 77] this would

be a mistake since we do not want the truth or falsity of an assertion to depend on interest (as indicated by posted goals) in achieving the assertion. As with goals this assertion is placed in the **Proposal-Justification-4** by the inheritance relation shown in the bottom right hand corner of the diagram. Thus we have one of the conjuncts of figure 9-5 established.

9.4. A Subgoal is Refuted

The attempt to establish the truth of the second conjunct follows in a similar manner. In this case the following rule is used to try to establish that an officer is a Schooled Officer for a particular billet:

$$(\Leftrightarrow (\text{for all } \equiv S$$
$$(\Rightarrow \equiv B \text{ is } (a \text{ Billet}$$
$$(\text{with unique Prereq-Schooling } \equiv S))$$
$$\equiv O \text{ is } (an \text{ Officer}$$
$$(\text{with Schooling } \equiv S))))$$
$$(is \equiv O (a \text{ Schooled-Officer}$$
$$(\text{with unique Billet } \equiv B))))$$

The difference is that in this case the outcome is the negation of the posted goal; Omega will assert that:

$$\neg (\text{OFFICER-6 is } (a \text{ Schooled-Officer}$$
$$(\text{with unique Billet BILLET-17})))$$

The failure to establish this fact implies the failure to establish the conjunctive goal in the rule on page 17 and hence the negation of the conjunction will be asserted which results in the negation of the second half of the equivalence:

$$\neg (\text{OFFICER-6 is } (a \text{ Qualified-Officer}$$
$$(\text{with unique Billet BILLET-17})))$$

We have been able to propagate back the fact that **OFFICER-6** was not a Schooled Officer because we had been using equivalences in our reasoning. When we get to our original implication, shown again below, we can go no further.

$$(\Rightarrow (\wedge \equiv B \text{ is } (a \text{ Billet-Fulfilling-Career-Objectives}$$
$$(\text{with unique Officer } \equiv O))$$
$$\equiv O \text{ is } (a \text{ Qualified-Officer}$$
$$(\text{with unique Billet } \equiv B)))$$
$$(is (an \text{ Officer-Billet-Proposal}$$
$$(\text{with unique Billet } \equiv B)$$
$$(\text{with unique Officer } \equiv O))$$
$$(a \text{ Reasonable-Proposal})))$$

This rule may be only one way that a proposal can be shown to be reasonable. There may be other rules that can possibly achieve the goal.

At this point the question is: how can we know when a goal cannot be achieved and how do we notify the

user. One approach is the following. Suppose there are only 3 conditions under which a proposal may be judged reasonable. The following rule could be used:

```
(⇔ (∨ r1 r2 r3)
    (is (an Officer-Billet-Proposal
            (with unique Billet ≡B)
            (with unique Officer ≡O))
         (a Reasonable-Proposal)))
```

Thus Omega can know that when all of r1, r2, and r3 fail then the goal cannot be established. This approach has two undesirable consequences. First, if the Assignment Officer asserts that a particular proposal is reasonable then Omega can conclude that one of r1, r2, or r3 is true which in fact may not be the case. There may be some other criteria that the Assignment Officer has used to judge a proposal as reasonable. The second problem is what to do when another criteria for judging a proposal reasonable is to be told to Omega. This would mean that the above rule would have to be contradicted and a new viewpoint would have to be constructed with an new equivalence rule with 4 criteria for judging a proposal reasonable.

9.5. Using Sponsors to Reason About Reasoning

A superior approach is to use information concerning the sponsor of a particular goal. As was described above, a sponsor is given a quanta with which to accomplish a goal. When the sponsor uses all its quanta it must ask for more to proceed. If a sponsor has quanta but can do no more work, i.e. it is quiescent, then it waits for additional work. The sponsor informs Omega about these events by making assertions. In our case the assertions will be simply the total quanta the sponsor has used. These assertions are made at two times, when the quanta allotted to the sponsor is exhausted or when the sponsor is quiescent.

Thus when a user posts a goal he or she will also specify the amount of quanta to be allocated to achieving the goal. When the quanta is used or no more of it can be used at a particular time then an assertion is made as to how much has been used. Note that if the assertion is made because the sponsor is quiescent at a particular time, this does not mean that no more can be used in the future. A new assertion, made from other sponsored activity, may once again enable work to be done on a particular goal. Thus in the case above when no more work can be done for a particular sponsor then the following is asserted.

```
Sponsor-1 is (a Quiescent-Sponsor
                (with Exhausted-Quanta 4.3))
```

Note that this assertion is monotonically compatible with past assertions of this type. The assertion will trigger a sprite that was created at the time the sponsor was given its quanta for the particular goal. Again, at the time the sprite actually fires it may well be that the sponsor is no longer quiescent. The sprite may well check to see if the sponsor is quiescent or if the sponsor's goal has been established. If the sponsor isn't quiescent or the goal has been established the sprite may take no further action. If the sponsor is quiescent

then it can examine the progress toward the goal. The progress toward the goal is analyzed by examining the DEPENDED-ON-BY attributes in the goal's justifications.

In this way Omega can determine what subgoals were posted for a goal and whether the goal or its negation was asserted. In our case it is determined that OFFICER-6 was determined not to be a qualified officer. The following information can be extracted from Omega's descriptions and presented to the user through a suitable user interface.

OFFICER-16 is not a Qualified Officer

and

OFFICER-16 is not Schooled and Experienced

An officer is Qualified if and only if he is Schooled and Experienced

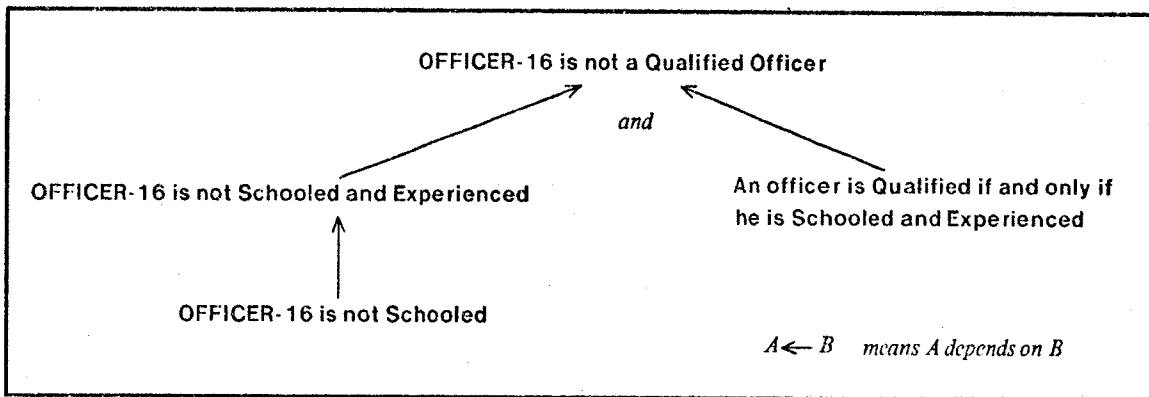OFFICER-16 is not Schooled

$A \leftarrow B$ means A depends on B

Figure 9-7: Why the Officer is not Qualified

Thus the user can see that the reason Omega has concluded that the officer is not qualified is because the officer is not Schooled. At this point the Assignment Officer may add the following assertion:

```
(⇒ (∧ ≡O is (an Experienced-Officer
                 (with unique Billet ≡B))
      ≡O is (a Schooled-or-Enrolled-Officer
                 (with unique Billet ≡B))
      ≡B is (a Billet-Fulfilling-Career-Objectives
                 (with unique Officer ≡O)))
   (is (an Officer-Billet-Proposal   ·
                 (with unique Billet ≡B)
                 (with unique Officer ≡O))
      (a Reasonable-Proposal)))
```

This assertion says that if an officer is experienced, (i.e., the officer is Schooled or enrolled in school) and if the billet satisfies the officer's career objectives then the proposal is a reasonable proposal. The officer would then go on to describe to Omega what it means for an officer to be schooled or enrolled in school for a particular billet.

10. Reasoning About Contradictions

In the previous section we have described how a user might interact with Omega when he or she is trying to achieve some goal and the goal cannot be achieved. The sponsors of a computation communicate with Omega and thus allow Omega to reason in a limited but useful fashion about the progress in achieving a particular goal. In this section we describe how contradictions are handled when they arise in the course of achieving some goal. For example, contradictions can arise when a user makes an assumption that violates a system constraint.

In the following example we will continue the scenario from the previous section of this chapter. Now the Assignment Officer has judged a proposal as reasonable and must calculate travel expenses for the proposed reassignment. The contradiction will arise when the Assignment Officer assumes there is enough money in the current quarter's expense account to cover the reassignment. To begin the calculation of travel expenses the Assignment Officer posts the goal that the proposal be financially viable:
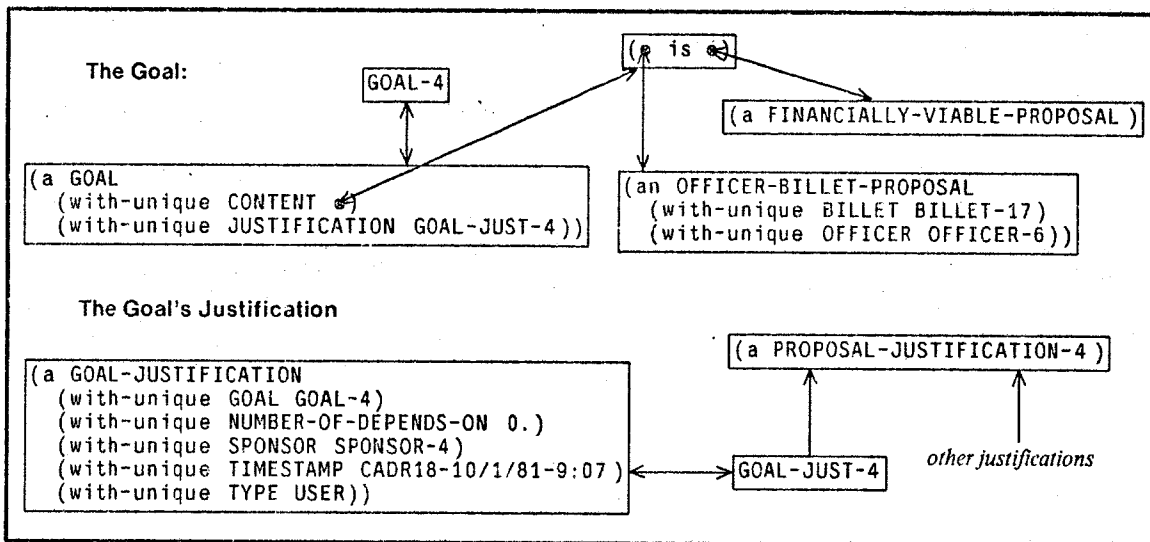

Figure 10-1: Representation of the Goal for Financial Viability

A sprite exists within Omega that watches for a goal of this sort. When the sprite fires on the goal it calculates the travel expenses for the proposed assignment and asserts this information. An abbreviated description of the sprite is shown below.

```
(when-goal Calc-Sprite-2 Travel-Expense-Sprite-Just-1
        (is (an 'Officer-Billet-Proposal                        ;Goal to match
            (with unique 'Officer ≡O)
            (with unique 'Billet ≡B))
        (a 'Financially-Viable-Proposal))
    ≡G-JUST ≡VP ≡SPONSOR                                        ;Goal elements

        1. Calculate travel expenses,
        2. Use current expense account,
        3. Assert the travel expenses and expense account.)
```

The assertion the sprite makes with its justification is shown in the diagram below.
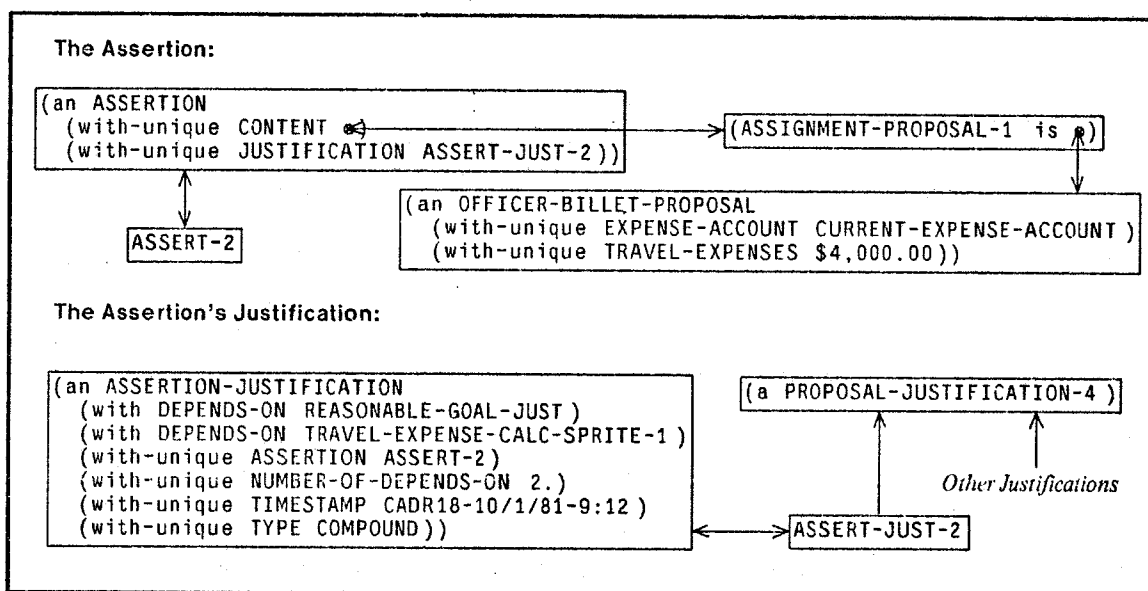

Figure 10-2: Travel Expense Assertion

The assignment proposal has been asserted to be *same* with the description Assignment-Proposal-1 for brevity. The above assertion states that the assignment proposal will incur a cost of $4,000.00 from the current expense account for travel expenses. Now Omega uses the following rule to calculate the new balance on the expense account.

```
(⇒ (is ≡P
       (an Officer-Billet-Proposal
           (with unique Travel-Expenses ≡TE)
           (with unique Expense-Account
                   (an Expense-Account
                       (with unique Account-# ≡AN)
                       (with unique Balance ≡B)))))
       (is (an Expense-Account (with unique Account-# ≡AN))
           (an Expense-Account
               (with unique New-Balance (- ≡B ≡TE)))))) [4]
```

Assume that the expense account has a balance of $1,000.00 and that the description of an expense account includes the following:

```
(an Expense-Account) is (an Expense-Account
                           (with every Balance (> = 0))
                           (with every New-Balance (> = 0)))
```

Where here we use the abbreviation (> = 0) for the description

```
(a Dollar-Amount (with Lesser-or-Equal-Amount 0)).
```

This descriptions describes an amount of dollars with the constraint that 0 is less than or equal to the amount.

When the rule that calculates the new balance fires it will assert that the new balance is $-3,000.00. This will be fused with the constraint that every balance and new balance be greater than or equal to 0. The attempt to fuse will fail, signalling a contradiction by making the following assertion:

---

[4] Note that we have used the abbreviation (- A B) for the description (a Difference (of Minuend A) (of Subtrahend B))

```
(a FAILURE-TO-FUSE-ATTRIBUTIONS
   (with PROBLEM-JUSTIFICATION  ASSERT-JUST-3 )
   (with PROBLEM-JUSTIFICATION  EXPENSE-ACCOUNT-DEFINITION-JUST )
   (with-unique ATTRIBUTE-NAME  NEW-BALANCE )
   (with-unique CONCEPT EXPENSE-ACOUNT )
   (with-unique NUMBER-OF-PROBLEM-JUSTIFICATIONS  2. ))
```

The Assertion of Contradiction:

```
                                                 (a CONTRADICTORY-VIEWPOINT
                                                    (with-unique REASON %))
(an ASSERTION
   (with-unique CONTENT %)
   (with-unique JUSTIFICATION CONTRADICTION-JUST-5 ))  (PROPOSAL-JUSTIFICATION-4  is %)
```

CONTRADICTION-ASSERT-5

The Justification:

```
(an ASSERTION-JUSTIFICATION                      (a PROPOSAL-JUSTIFICATION-4 )
   (with DEPENDS-ON  REASONABLE-GOAL-JUST )
   (with DEPENDS-ON  TRAVEL-EXPENSE-CALC-SPRITE-1 )
   (with-unique ASSERTION ASSERT-2 )
   (with-unique NUMBER-OF-DEPENDS-ON  2. )
   (with-unique TIMESTAMP CADR18-10/1/81-9:12 )
   (with-unique TYPE COMPOUND ))               ASSERT-JUST-2
```
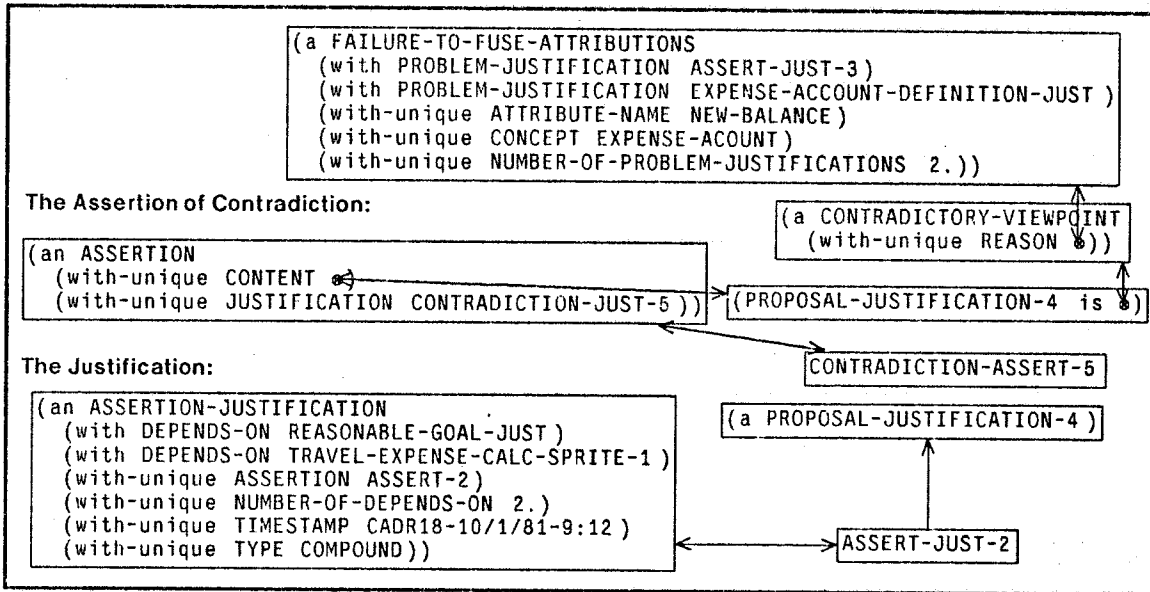
Figure 10-3: Assertion of the Contradiction

In the above, we have assumed that the assertion which calculated the NEW-BALANCE has the justification ASSERT-JUST-3. A sprite will fire when the contradiction is asserted. The sprite will retrieve the justifications for the offending assertions. The sprite will analyze the assertions, retrieving the descriptions in the NEW-BALANCE attributions and present the following information to the user:
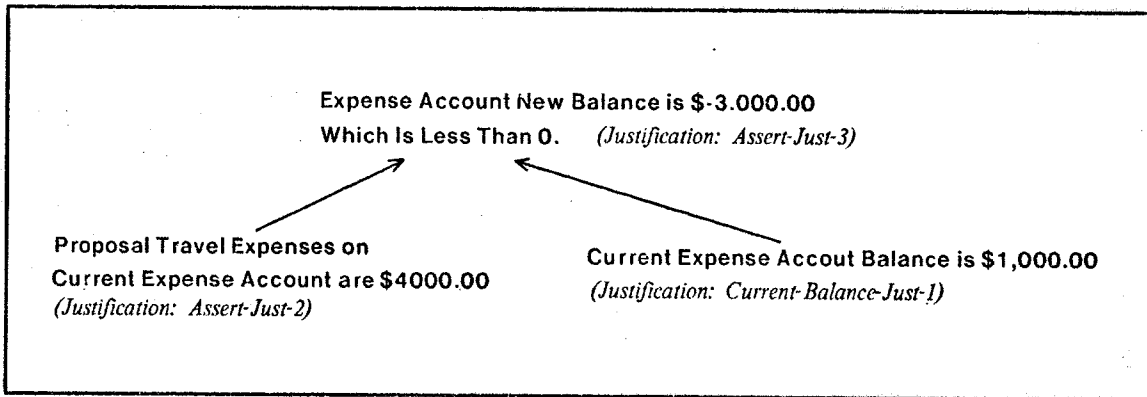
Expense Account New Balance is $-3.000.00
Which Is Less Than 0.   *(Justification: Assert-Just-3)*

Proposal Travel Expenses on
Current Expense Account are $4000.00
(Justification: Assert-Just-2)

Current Expense Accout Balance is $1,000.00
(Justification: Current-Balance-Just-1)

Figure 10-4: Contradiction Dependencies

When this information is presented, the user can immediately see the cause of the contradiction and can take corrective actions, for example use the next quarters expense account.

11. Conclusion

We have presented the Omega knowledge system along with some simple examples. Omega's problem solving and viewpoint capabilities are useful for describing and reasoning about objects whose properties vary with time and for handling contradictions that arise during reasoning processes. The examples are simple have been chosen to illustrate Omega's functionality. They do not reflect the level of complexity that Omega can handle. In the first example Omega performed part of an office procedure after an office worker posted the goal of the procedure. In an actual application Omega could perform many complicated tasks relating to any particular goal that an office worker might post. In the second example Omega discovered and illustrated a problem in an attempt to establish a goal. In actual applications this would be very useful as obstacles in the performance of complicated tasks could be easily found with Omega's aid.

The viewpoint mechanism presented here is related to that in ETHER [Kornfeld 79] and to the layers of the PIE system [Goldstein 80]. Viewpoints are a powerful unifying mechanism which combine aspects of McCarthy's situational tags [McCarthy, Hayes 69] and the contexts of QA4 [Rulifson, Derksen, Waldinger 72]. They serve as a replacement for update and pusher-puller mechanisms.

Office work is a problem solving activity that can be supported with the use of knowledge embedding languages. A Problem Solving Support Paradigm has been presented as a framework within which to develop support tools for the office worker. The Omega Viewpoint mechanism provides a facility to help the office worker achieve his or her goals, and to analyze contradictions when they arise.

12. Acknowledgments

I would like to thank Carl Hewitt, my research advisor, for sparking my interest in this field. He has been an invaluable source of inspiration and insightful questions. Almost all of the work presented in this paper is based on his ideas. This research has been influenced by conversation with many people. For listening and commenting on the work I would like to thank Giuseppe Attardi, Mike Brady, Marilyn Brockman, Gene Ciccarelli, Stan Curtis, Bill Kornfeld, Henry Lieberman, Maria Simi, and John Teeter. The diagrams in this paper were made with the Omega Presenter on MIT's Lisp Machine. The presenter was developed by Gene Ciccarelli in conjunction with the author.

13. References

[Barber 82]
    Barber, G. R.
    Office Semantics.
    PhD thesis, Massachusetts Institute of Technology, 1982.

[de Kleer, Doyle, Steele, Sussman 77]
de Kleer, J., Doyle, J., Steele, G. L., and Sussman, G. J.
AMORD: Explicit Control of Reasoning.
In *Proceedings of the Symposium on Artificial Intellignece and Programming Languagues.* SIGART, Rochester, N.Y., August, 1977.

[Doyle 77]
Doyle, J.
Truth Maintenance Systems for Problem Solving.
AI-TR 419, MIT AI Lab, May, 1977.

[Ellis, Nutt 80]
Ellis, C. A. and Nutt, G. J.
Computer Science and Office information Systems.
Computing Surveys 12(1), March, 1980.

[Fikes, Henderson 80]
Fikes, R. E. and Henderson, D. A.
On Supporting the Use of Procedures in Office Work.
In *Proceedings of the First Annual AAAI Conference.* American Association for Artificial Intelligence, August, 1980.

[Fikes 80]
Fikes, R. E.
Odyssey: A Knowledge-Based Assistant.
To appear in Artificial Intelligence.

[Goldstein, Roberts 77]
Goldstein, I. P. and Roberts, R.B.
NUDGE, a Knowledge-Based Scheduling Program.
Proceedings of the Fifth International Joint Conference on Artificial Intelligence.

[Goldstein 80]
Goldstein, Ira.
PIE: A Network-Based Personal Information Environment.
In *Proceedings of the First Annual AAAI Conference.* American Association for Artificial Intelligence, August, 1980.

[Kornfeld 79]
Kornfeld, W.
Using Parallel Processing for Problem Solving.
AI Memo 561, MIT, December, 1979.

[Kornfeld 82]
Kornfeld, W.
Concepts in Parallel Problem Solving.
PhD thesis, Massachusetts Institute of Technology, 1982.

[McCarthy, Hayes 69]
> McCarthy, J. and Hayes, P. J.
> Some Philosophical Problems from the Standpoint of Artificial Intelligence.
> In *Machine Intelligence 4*, pages 463-502. Edinburgh University Press, 1969.

[Rulifson, Derksen, Waldinger 72]
> Rulifson, J., Derksen, J. and Waldinger, R.
> *QA4: A Procedural Calculus for Intuitive Reasoning.*
> Artificial Intelligence Center Technical Note 73, Stanford Research Institute, November, 1972.

[Steele 80]
> Steele, G. L.
> *The Definition and Implementation of a Computer Programming Language Based on Constraints.*
> AI TR 595, MIT AI Lab, August, 1980.

[Suchman 79]
> Suchman, L.
> *Office Procedures as Practical Action: A Case Study.*
> Technical Report, XEROX PARC, September, 1979.

[Sussman 72]
> Sussman, G. J.
> From PLANNER to CONNIVER - a Genetic Approach.
> In *Proceedings of the AFIPS Fall Joint Computer Conference.* AFIPS, 1972.

[Tsichritzis 79]
> Tsichritzis, D.
> A Form Manipulation System.
> In *Proceedings of the NYU Symposium on Automated Office Systems.* NYU, 1979.

[Winograd 71]
> Winograd T.
> *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language.*
> MAC TR 83, MIT, 1971.

[Wynn 79]
> Wynn, E.
> *Office Conversation as an Information Medium.*
> PhD thesis, Department of Anthropology, University of California, Berkeley, 1979.

[Zloof 80]
> Zloof, M. M.
> *A Language for Office and Business Automation.*
> Research Report 35086, IBM, January, 1980.