MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 641 January, 1981

The Scientific Community Metaphor

William A. Kornfeld and Carl Hewitt

ABSTRACT. Scientific communities have proven to be extremely successful at solving problems. They are inherently parallel systems and their macroscopic nature makes them amenable to careful study. In this paper the character of scientific research is examined drawing on sources in the philosophy and history of science. We maintain that the success of scientific research depends critically on its concurrency and pluralism. A variant of the language Ether is developed that embodies notions of concurrency necessary to emulate some of the problem solving behavior of scientific communities. Capabilities of scientific communities are discussed in parallel with simplified models of these capabilities in this language.

ACKNOWLEDGEMENTS: This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Office of Naval Research under Office of Naval Research contract N00014-75-C-0522.

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-11, NO. 1, JANUARY 1981

The Scientific Community Metaphor

WILLIAM A. KORNFELD AND CARL E. HEWITT

Abstract—**Scientific communities have proven to be extremely successful at solving problems. They are inherently parallel systems and their macroscopic nature makes them amenable to careful study. In this paper the character of scientific research is examined drawing on sources in the philosophy and history of science. We maintain that the success of scientific research depends critically on its concurrency and pluralism. A variant of the language Ether is developed that embodies notions of concurrency necessary to emulate some of the problem solving behavior of scientific communities. Capabilities of scientific communities are discussed in parallel with simplified models of these capabilities in this language.**

I. Introduction

MUCH OF the present interest in parallel processing stems from a now well-documented technological trend—the declining cost of processing, storage, and communications. This is assuredly a valid motivation, but not the only one. Our research in artificial intelligence has led us to a very different reason for considering concurrent systems. We now feel parallelism is fundamental to the design and implementation of expert systems in many domains. In this paper we present a largely philosophical discussion of our motivations. A language called Ether has been designed to create highly parallel problem solving systems. The actual language differs in several details from the one described in this paper which has been created for pedagogical reasons. A discussion of the Ether language can be found in [11].

Engineers wishing to construct an artifact capable of implementing a process, such as problem solving, often study naturally occurring systems that already implement the process. This approach has led many researchers to investigate psychological models of human thought as a basis for constructing an artificial intelligence. Some of this research is reviewed later. Here we focus our attention on how scientific communities solve problems. That scientific communities are successful at generating and deciding between alternative explanations for phenomena is indisputable. Scientific progress, looked at globally and with a time scale of many decades, seems coherent and purposeful. Looked at locally, this is anything but true. At any one time many conflicting theories may purport to explain the same phenomenon. Scientists within a field often engage in highly charged arguments with one another. Occasionally what may ultimately turn out to be the wrong party to a dispute will gain temporary popularity; though the fields themselves seem to grow in depth and power over the long haul. We believe the overall success of scientific research is due in large part to its tolerance of this diversity.

Scientific communities themselves can be the subject matter of scientific research. The nature of science has been a fertile topic in philosophy from the pre-Socratics through the present day. We hope to gain useful insights from this research that will aid us in the design of computer systems with some of the capabilities of scientific communities. We are particularly indebted to a number of philosophers and historians of science of this century, among them Popper, Lakatos, Kuhn, and Feyerabend. The observations expressed in the following sections owe a great deal to their work.

Our thesis is that the structure of problem solving in scientific communities can be used to justify many of the design decisions for parallelism and pluralism in Ether. We discuss the constructs of Ether in parallel with the characteristics of scientific communities that motivate them. We hope to glean useful ideas from the metaphor of scientific research that will aid in the future development of problem solving systems based on Ether.

We wish to warn the reader not to look for a direct correspondence between the components of Ether and analogous components in scientific communities. The relationship is one that can be seen only at a high level of abstraction. The most important aspects of this relationship are the high degree of concurrency and the nature of the concurrent activities with respect to the overall problem solving effort.

II. Communication

One of the most salient aspects of scientific communities is that they are *highly parallel systems*. Scientists work on problems concurrently with other scientists. They can work on the same problem or on quite different problems. They may or may not know of the work of other scientists. They may hold similar opinions to one another or quite divergent opinions. Popper and Lakatos have developed the thesis that this diversity is a *necessary* aspect of successful scientific research. One of our goals in the development of Ether is to facilitate the construction of systems with this kind of diversity.

Scientists do not, of course, work in a vacuum. They are able to communicate their ideas with one another. One scientist's results can sometimes have a profound effect on the nature of future work by other scientists. A principle goal in the design of the Ether language is making interaction between different parts of the system easy without

Manuscript received February 10, 1980; revised August 10, 1980.

The authors are with the Department of Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

many of the pitfalls that make current parallel processing systems, based on mechanisms like semaphores and shared memory, difficult to use.

Some of the communication in scientific communities concerns *resource control*. Research programs that seem more likely to lead to solutions of the goals of the system are allocated more resources than their less likely alternatives. Some of the activity of an Ether system is concerned with comparing competing methodologies and reallocating resources on the basis of relative merit.

All communication in Ether is done by *disseminating* messages. There are several kinds of communication that might take place. Scientists often communicate *results* of their own researches. We will refer to these kinds of messages as *assertions*. There are other kinds of messages that must be communicated. At any time the system has certain *goals*, either to demonstrate the validity of a proposition or to find a method for solving a given problem. The goals of one part of the system must be communicated to parts of the system embodying expertise that can help achieve the goal. This communication is done with messages.

Computation in Ether is done by computational elements known as *sprites*. For each sprite s, there is a set of messages called (InterestSet s) in which the sprite is potentially interested. Sprites can communicate with each other by disseminating messages. If a message m is disseminated which is an element of (InterestSet s_k), then s_k will receive the message m. The intent of dissemination is to achieve the effect of broadcasting a message without incurring all of the overhead of broadcasting.

As a result of receiving a message in which it is interested, a sprite can create new sprites and send more messages. A running Ether program contains a set of sprites and disseminated messages that interact with each other to produce more sprites and messages. Dissemination in Ether has important properties called *monotonicity, commutativity, parallelism,* and *pluralism* which are explained below.

Monotonicity: Once a message m is disseminated it cannot be "erased." It will remain available forever.

Commutativity: If a message m is an element of (InterestSet s) of a sprite s, then s will receive m regardless of whether the message was disseminated before or after the sprite was activated.

Parallelism: If a message m is disseminated which is in the interest set of sprites $s1$ and $s2$, then $s1$ and $s2$ will process the message concurrently. Similarly, if messages $m1$ and $m2$ are disseminated which are both in the interest set of a sprite s, then s will process the messages concurrently. More generally, if two Ether subsystems (collections of sprites and messages) $E1$ and $E2$ will produce larger collections of sprites and messages $E1'$ and $E2'$ when run in isolation from one another, then an Ether system consisting of $E1 \cup E2$ will produce a system E when run such that $E1' \cup E2' \subseteq E$.

Pluralism: Ether supports working on multiple and not necessarily compatible hypotheses at the same time.

The above properties are idealizations of certain characteristics of scientific communities. They allow computer programs to be written that emulate the way scientists work together in cooperation and competition.

We will now attempt to make more explicit the correspondence between the role played by these properties in Ether and their role in scientific communities.

Monotonicity: Scientists *publish* their results so they are available to all who are interested. Published work is collected and indexed in libraries. Scientists who change their mind can publish a later article contradicting the first. However, they are not allowed to go into the libraries and "erase" the old publication. Publications advocating obsolete and unpopular theories are stored along with currently popular theories.

Commutativity: It does not matter whether a scientist becomes interested in a publication before it is published or vice versa for it to be of use. Scientists who become interested in a scientific question make an effort to find out if the answer has already been published. In addition they attempt to keep abreast of further developments as they continue their work.

Parallelism: Scientists can work concurrently with one another without adverse effects.

Pluralism: There is no central arbiter of truth in scientific communities.

The properties of monotonicity and commutativity are goals of the scientific community which are only incompletely achieved in practice. For example, a publication can be lost or a scientist may not have read or understood an article important to his own research. Furthermore, in scientific communities the goals of commutativity and parallelism are tempered by resource constraints; it may be easier to perform an experiment than to determine if it has already be done. Ether has mechanisms for resource control that are intended to address some of these issues.

A. *An Example in Ether*

Sprites can communicate with other sprites by disseminating messages. A sprite consists of two parts, a *trigger pattern* and a *body*. The trigger pattern of a sprite s is a convenient way to express (InterestSet s), the set of messages in which the sprite is interested. When information has been disseminated that matches this template, the body of the sprite (Ether code) is executed in an environment supplied by the match. There may be code in the body to create new sprites or disseminate information to other sprites.

To make the concept of sprites more understandable it is useful to treat an example in detail. We will consider the problem of determining whether or not a path exists in a directed graph from one set of nodes called *origins* to another called *destinations*. Problems of this kind are common in problem solving and planning situations. The origins might represent the current situation or the hypotheses of the theorem to be proved. The destinations would represent the desired situation or the conclusions of the theorem.

The particular problem to be solved is posed to the system by disseminating a message of the form (GOAL (PathExists Origins Destination)). The goal is recorded as being achieved by disseminating a message of the form (ASSERTION (PathExists Origins Destinations)).

The data base contains messages indicating patterns of local connectedness in the graph. A message of the form (ImmediatePredecessor ⟨N1⟩⟨N2⟩) signifies that node ⟨N1⟩ is known to be an immediate predecessor of node ⟨N2⟩. Similarly, a message of the form (ImmediateSuccessor ⟨N1⟩⟨N2⟩) indicates ⟨N1⟩ immediately succeeds ⟨N2⟩.

An expression of the form

```
(WHEN ⟨TRIGGER⟩⟨COMMAND_1⟩ ··· ⟨COMMAND_j⟩)
```

is used to create a new sprite ⟨s⟩ with a trigger pattern ⟨TRIGGER⟩ and j commands. An expression of the form (ACTIVATE ⟨s⟩) can be used to activate the sprite ⟨s⟩. Thereafter, when messages which have been disseminated match the trigger, the commands are executed in an environment provided by the match. The commands are executed concurrently. These expressions are illustrated in the following definition of the procedure AlwaysChainBothWays which has no parameters:

```
(DEFINE (AlwaysChainBothWays)
  (ACTIVATE
    (WHEN (GOAL (PathExists = U = V))
      (ChainForward U V)
      (ChainBackward U V)))).
```

Suppose the expression (AlwaysChainBothWays) has been executed (activating a sprite we will call s) and that the message

```
(GOAL (PathExists Origins Destinations))
```

(which we will call m) has been disseminated. Then regardless of whether s is activated before or after m is disseminated, the sprite s will receive the message m and will bind the identifier U to Origins and the identifier V to Destinations. The following two commands will then be executed concurrently:

```
(ChainForward Origins Destinations)
(ChainBackward Origins Destinations).
```

A command of the form (AlwaysNoticeTrivialPaths) will look to see if the sets ⟨X⟩ and ⟨Y⟩ have any nodes in common when a goal of the form (PathExists ⟨X⟩⟨Y⟩) is disseminated. If this is the case an assertion to that effect will be disseminated. The following definition accomplishes the desired result:

```
(DEFINE (AlwaysNoticeTrivialPaths)
  (ACTIVATE (WHEN (GOAL (PathExists = X = Y))
    (IF ((X ∩ Y) ≠ φ)
      THEN (DISSEMINATE
        (ASSERTION (PathExistsX Y))))))).
```

A command of the form (ChainForward ⟨X⟩⟨Y⟩) will cause a subgoal for the form (PathExists ⟨s⟩⟨Y⟩) to be disseminated for each successor ⟨s⟩ of ⟨X⟩. Furthermore, for each successor ⟨s⟩, a new sprite is created which disseminates the assertion that a path exists from ⟨X⟩ to ⟨Y⟩ if the subgoal is achieved.

```
(DEFINE (ChainForward = X = Y)
  (WHEN (ImmediateSuccessor = s X)
    (DISSEMINATE (GOAL (PathExists {s} Y)))
    (ACTIVATE (WHEN (ASSERTION (PathExists {s} Y))
      (DISSEMINATE (ASSERTION (PathExists X Y))))))).
```

Before continuing with this example, we would like to introduce some abbreviations which will reduce the bulk of the code without changing its meaning. Commands of the form (DISSEMINATE (ASSERTION ⟨x⟩)) are very common in the above code. We introduce the following definition so that we can use the abbreviation (ASSERT ⟨x⟩) instead.

```
(DEFINE (ASSERT=x)
  (DISSEMINATE (ASSERTION x))).
```

Additionally, command fragments of the form

```
(DISSEMINATE (GOAL ⟨g⟩))
(ACTIVATE (WHEN (ASSERTION ⟨g⟩)
  ⟨COMMAND_1⟩
  ...
  ⟨COMMAND_k⟩))
```

will be abbreviated as follows:

```
(SHOW ⟨g⟩
  (WhenAchieved
    ⟨COMMAND_1⟩
    ...
    ⟨COMMAND_k⟩)).
```

Using these abbreviations, the code for ChainForward appears as follows:

```
(DEFINE (ChainForward = X = Y)
  (WHEN (ImmediateSuccessor = sX)
    (SHOW (PathExists {s} Y)
      (WhenAchieved
        (ASSERT (PathExists XY)))))).
```

Commands of the form (chainBackward ⟨X⟩⟨Y⟩) are completely analogous to the ones for chaining forward. The corresponding definition is given below:

```
(DEFINE (ChainBackward = X = Y)
  (WHEN (ImmediatePredecessor = p Y)
    (SHOW (PathExists X {p})
      (WhenAchieved
        (ASSERT (PathExists X Y)))))).
```

This system is set in motion by executing the commands

```
(AlwaysChainBothWays)
```

and

```
(AlwaysNoticeTrivialPaths).
```

In some examples introduced late in this paper we will find it useful to have a concise notation for a sprite which triggers on a set of messages which have been disseminated. We will use the notation

```
(WHEN {⟨PAT₁⟩⟨PAT₂⟩}
  ⟨COMMAND₁⟩
  . . .
  ⟨COMMANDⱼ⟩)
```

for a sprite which triggers on a set of messages which match ⟨PAT₁⟩ and ⟨PAT₂⟩. The above notation can be regarded as an abbreviation for the presence of either:

```
(WHEN ⟨PAT₁⟩
  (ACTIVATE (WHEN ⟨PAT₂⟩
    ⟨COMMAND₁⟩
    . . .
    ⟨COMMANDⱼ⟩)))
(WHEN ⟨PAT₂⟩
  (ACTIVATE (WHEN ⟨PAT₁⟩
    ⟨COMMAND₁⟩
    . . .
    ⟨COMMANDⱼ⟩))).
```

We reason that if messages M_1 and M_2 are disseminated and they match ⟨PAT₁⟩ and ⟨PAT₂⟩ respectively, then the commands ⟨COMMAND₁⟩,···,⟨COMMANDⱼ⟩ will be executed with the same bindings in both cases.

III. Proposers

Scientific research consists of generating proposals to account for observations and processes for substantiating and refuting these theories. Popper has called the process *conjecture and refutation.* Theories evolve and become substantiated or rejected through an interplay of many processes. Three basic types are proposers, proponents, and skeptics. Proposals are new theories, goals, and techniques put forward for critical assessment and development. Proponent activities attempt to substantiate these proposals. Skeptical activities play the role of "devil's advocate" to test proposals using any techniques the system has at its disposal.

An important source of proposals are very general methods coming from "common sense" knowledge. For example, when trying to get from X to Z, it is often desirable to find another place Y such that Z is accessible from Y and Y is accessible from X. Of course these simple solutions will often not work without further elaboration. In this particular example it may be that it is better to overcome the obstacle which prevents direct access from X to Z than to try to find another route. For example suppose that X and Z are fields and there is a fence between them. It may be better to try to find a way through the fence than to find another field Y which is accessible to X and Z. We believe "common sense knowledge" is the source of many general proposal generators of this kind.

A somewhat more abstract way to generate new hypotheses is by metaphor. A situation may have several features in common with another situation for which a solution is known to be successful. Metaphor can be used to explain the origins of the two alternative theories of light debated early in the nineteenth century. Newton suggested that light consists of particles because it shares several features in common with particles. With no obstructions light moves in a straight line. If a mirror is placed in its path the light will be reflected at the same angle as would particles in an elastic collision with the surface of the mirror. Newton was very familiar with the concept of a particle. There were enough similarities between the theory of particle interaction and observed characters of light to put forward the theory that "light consists of particles."

Theories are proposed and then tested. We use the term "skeptics" for activities whose purpose is to reason about the implications of theories looking for *anomalies*. Anomalies are points where the implications of a theory differ from prediction. In many cases a theory is modified to account for the anomaly, but in a way that preserves the character and support structure of the original theory. This process is called *adjustment* (following Lakatos) and is discussed later in this paper.

IV. Proponent Activities

When new goals are proposed, sprites go to work attempting to establish these goals. Activities attempting to achieve a goal are collectively known as *proponents.* We have already presented an example of the efficacy of multiple proponents attempting to achieve a goal. When the message (GOAL (PathExists Origins Destinations)) is disseminated many sprites go to work trying to establish it. The activity of these sprites constitutes the proponent activity for the goal.

The most common kind of proponent we have studied thus far is what is sometimes known as *working backward from the goal* or *consequent reasoning*. The goal can be established by establishing one or more subgoals. The proponent functions by proposing these new goals.

A. Disjunctive Subgoals

For example, if we wanted to establish that someone (say "Joe") was a U.S. citizen, it would suffice to show he was either born in the U.S. or he was naturalized. To do this in Ether we create a sprite that watches for messages of the form

```
(GOAL (USCitizen ⟨PERSON⟩))
```

and then establishes new goals

```
(GOAL (NativeBorn ⟨PERSON⟩))
```

and

```
(GOAL (NaturalizedUSCitizen ⟨PERSON⟩)).
```

If either of the proponents working on these new goals succeeds, we are justified in asserting

```
(USCitizen ⟨PERSON⟩).
```

This can be said in Ether in the following way:

```
(WHEN (GOAL (USCITIZEN = PERSON))
   (SHOW (NATIVEBORN PERSON)
      (WHENACHIEVED
         (ASSERT (USCITIZEN PERSON))))
   (SHOW (NATURALIZEDUSCITIZEN PERSON)
      (WHENACHIEVED
         (ASSERT (USCITIZEN PERSON))))).
```

When a goal that matches (USCITIZEN = PERSON) is received by the above sprite it disseminates the two new subgoals

(NATIVEBORN PERSON)

and

(NATURALIZEDUSCITIZEN PERSON).

Additionally it activates sprites that watch for either of the two new goals to be achieved. If either is established the respective sprite asserts (USCITIZEN PERSON) to record that the goal (which triggered the whole sprite) has been achieved.

B. Conjunctive Subgoals

The previous example was of a situation where *any* of the subgoals could be achieved in order to achieve the initial goal. There is an analogous kind of proponent for which *all* subgoals must be achieved for the main goal to be achieved. To exemplify this, suppose we wish to establish that a person is capable of becoming naturalized as a U.S. Citizen. There are two conditions that must be shown; he must have lived in the U.S. for at least 5 years *and* has knowledge of U.S. government. We can implement this proponent activity with the following sprite:

```
(WHEN (GOAL (CANBENATURALIZED = PERSON))
   (SHOW {(> (YEARSRESIDENT PERSON) 5)
      (KNOWLEDGEABLE PERSON USGOVERNMENT)}
         (WHENACHIEVED
            (ASSERT (CANBENATURALIZED PERSON))))).
```

V. SKEPTICAL ACTIVITIES

Scientific knowledge is perpetually in a state of evolution. *All* scientific theories and beliefs are subject to overthrow. This realization is a relatively recent one in Western thought. The spectacular achievement of relativistic mechanics played a large role in changing our opinions of the significance of scientific knowledge. Newtonian mechanics had achieved the status of incontrovertible truth in the minds of most thinkers until the beginning of this century. The discovery that even this superbly justified theory could be ultimately found false left little room for complacency about the ultimate status of any beliefs.

Popper's philosophy of science begins with the observation that an asymmetry exists between the logical provability and refutability of scientific theories. Observation statements can never logically imply such theories but they *can* logically refute them. A simple example of this is the hypothesis "All swans are white." No matter how many white swans we observe we can never deduce on logical grounds the truth of this theory. However, from an observation statement reporting just one black swan, we can logically conclude that "All swans are white." is false.

The simple theory implied by the white swan example is an obvious oversimplification. Its importance in the historical development of the philosophy of science is the beginning of the doctrine of *falsificationism.* The name Lakatos [14] gives for the most elementary doctrine, in which a single refuting "observation" would disallow the entire theory, is *naive falsificationism.* More sophisticated theories have emerged which will be discussed shortly. We call activities whose purpose is to discover anomalous implicants of theories *skeptics.*

The efforts of skeptics is then largely applied in attempts to discredit scientific conjectures. Popper proposed that it is essential to the progress of science that many scientists hold conflicting explanations of phenomena. Scientists spend must of their time deducing the implications of theories (theirs or others) and testing the agreement of these implications with observations. The acceptability of a theory or explanation is related to its ability to avoid being discredited. This is the idea of natural selection applied to theories. The polynomial is not equal to nondeterministic polynomial ($P \neq NP$) conjecture provides a convincing example from contemporary computer science. There is no *logical* reason for believing that no deterministic algorithms for NP-complete problems run in polynomial time. Yet people lay great faith in the truth of this statement. Why such certainty in a mere conjecture? It is commonly believed because *so many people have tried to disprove it and failed.*

Abandoning a theory whenever an anomaly appears would lead one to reject every scientific theory ever proposed. Few scientific theories of any worth are so pristine that they can be said to entail no anomalous implications. The anomalies engendered by a scientific theory must be weighed in the context of ongoing research on alternative theories. These anomalies will often lead to modifications of the theories in a process known as *adjustment* to be discussed shortly.

VI. SPONSORS

A fact about real computers as well as the society in which we live is that resources are finite and therefore must be allocated to what we consider to be the important tasks at hand. This is reflected in Ether through mechanisms to allocate the available computational resources to the activated sprites. The scientific community has several mechanisms for allocating effort. Two important ones are funding structures and peer review. Scientific research that is considered presently useful is more likely to get funded and thus pursued. Similarly, a scientist who investigates theories in current vogue is more likely to achieve the admiration of his peers, get university posts, etc., that will promote his further research.

To model this process in Ether, we must find some way of allocating computational effort to goals. We call an agent that provides computational effort a "sponsor." All work done happens under some sponsor; in particular, all sprites capable of triggering do so through the support of a sponsor. Trying to disprove the flat-earth theory is an example of a goal which may (or most likely, may not) be supported by a sponsor.

When the potential results of some proposed activity in support of a goal do not seem of sufficient value, the sponsor will not provide much (or any) resources. Conversely, computations which seem more promising to the sponsor are given greater resources. Often several competing approaches will be possible in a given situation. The system, not having enough knowledge to conclude categorically that one will succeed over another, will apportion resources to several. A good example of this in today's world is the search for a safe and abundant energy source. There are those who argue that nuclear power is the only viable near-term energy source, those who argue for solar power, and those who favor coal. While one of these groups of people may very well be right, the system as a whole supports activity working on all. Society as a whole has little alternative but to patiently wait for the results of one group to outstrip the others.

Goals are created with sponsors that support the activities (both proponents and skeptics). If the goal has been achieved, or if it is demonstrated that the goal cannot be achieved, all the activity can be halted by making a request to the sponsor. In Ether a command of the form (WITHHOLD ⟨s⟩ (WITH REASON ⟨r⟩)) can be used to request that the sponsor ⟨s⟩ withhold further support for the reason ⟨r⟩.

In a previous section we showed how to we could use sprites to begin work on a goal. In the example we had a goal of determining whether Joe could be naturalized. We will expand our treatment of this example to illustrate the use of sponsors.

To assign goals to sponsors we expand the message disseminated to be

```
(GOAL (CanBeNaturalized Joe) (WITH SPONSOR =s)).
```

In addition to containing an indication of the goal

```
(CanBeNaturalized Joe)
```

the message states that *s* will sponsor work aimed at achieving it. If it is determined that Joe has lived in the U.S. for less than five years then we must give up all hope of showing that he can be naturalized. This is one kind of skeptic for the goal (CanBeNaturalized Joe) is shown in the following:

```
(WHEN {(ASSERTION (YearsResident Joe =n))
  (ASSERTION (≤ n 5))}
    (WITHHOLD s (WITH REASON
      {(ASSERTION (YearsResident Joe =n))
        (ASSERTION (≤ n 5))}))).
```

If an assertion of the form (YearsResident Joe 2) is present, it will cause the sponsor *s* to withhold support for the goal. A skeptic has succeeded in showing a goal, on which the system has been investing some of its resources, is unattainable. These resources can then be given to more promising ventures.

Another way sponsors are used in the system is to prevent resources from being wasted attempting to establish *results already known*. The example of a proponent to achieve the goal of showing that a person can be naturalized has been reformulated below to show the use of sponsors and skeptics:

```
(WHEN (GOAL (CanBeNaturalized =PERSON)
  (WITH SPONSOR =s))
    (SHOW {(>(YearsResident PERSON) 5)
      (Knowledgeable PERSON USGovernment)}
        (WhenAchieved
          (ASSERT (CanBeNaturalized PERSON))
          (WITHHOLD s (WITH REASON
            (ESTABLISHED
              (CanBeNaturalized PERSON))))))
    (SHOW (NOT (>(YearsResident PERSON) 5))
      (WhenAchieved
        (WITHHOLD s (WITH REASON
          (NOT (>(YearsResident PERSON) 5))))))
    (SHOW (NOT
      (Knowledgeable PERSON USGovernment))
        (WhenAchieved
          (WITHHOLD s (WITH REASON
            (NOT (Knowledgeable PERSON
              USGovernment))))))).
```

The first *show* command disseminates two subgoals. If they are both achieved for some person then the sponsor is requested to withhold support for working on the goal for that person. The second *show* command disseminates a goal to establish that the person has been a resident in the U.S. less than five years; if established the sponsor *s* is asked to withhold support for the goal. The last *show* command creates a similar goal to show that the person is not knowledgeable about U.S. Government.

We believe that the construction of truly complex problem solving systems will require more precise mechanisms of control than solely the abilities to start and stop work on a goal; it should be possible to let activities use different amounts of resources in accordance with the system's assessments of how useful the results are likely to be. Sponsors are allowed to assign varying amounts of *processing power*, measured in units of cycles per second, to activities. Sponsors, when created, are allocated a certain quantity of processing power which they can in turn allocate to sponsors they create. A sponsor can redistribute any resources it has been allocated at any time as new knowledge is gained.

The statement of the properties of monotonicity, commutativity, and parallelism must be slightly rephrased in order to be valid in systems with a sponsor because a message *m* will cause the execution of the body of a sprite *s* only if *s* can obtain resources from a sponsor.

VII. Adjustment

Conjectures often develop by adjusting to anomalies discovered in their predecessors. Research programs are often manifest as producing a sequence of theories. Each one is an adjustment of its predecessor and is able to handle more cases than its predecessor. Lakatos [13] illustrates this through a historical development of the Euler formula for the relationship between the number of faces, vertices, and edges of a polyhedron. Euler's formula is $F + V = E + 2$. Lakatos then presents counterexamples such as an object which is a cube with a smaller cube glued to one of its sides as shown in Fig. 1. This object has 11 faces, 16 vertices, and 24 edges. When this anomaly was discovered, an examination of the proof of Euler's formula led to the discovery that the discrepancy between the proven theorem and the counterexample could be attributed to the ring-shaped face shown shaded in the diagram. The process lead to better understanding of the Euler's formula and the concept of "polyhedron." Lakatos analyzes some of the techniques by which mathematical hypotheses are modified in the face of such refutations.

When an anomaly is discovered in a theory, the outcome is often an adjustment of the theory rather than a total dismissal. This is because theories that have shown some success tend to develop a core of fundamental concepts that serve to motivate further work. The process of adjustment, then, is an effort to protect this core from being discredited. Kuhn has pointed out the ubiquity of these cores of successful research programs (that he calls *paradigms*) in the sciences. The significance of paradigms in the scientific communities is that they provide the framework of research programs to investigate promising ideas.

Only the most trivial kinds of theories can be proposed correctly the first time. Most theories that gain significant acceptance evolve through a continual interplay of proponents, skeptics, and adjustment. This aspect of the workings of a scientific community has not yet been modeled in an interesting way in Ether.

A. Adherence

We can make several important observations about the characterization of *adherence* to positions, theories, methods, etc. in scientific communities. We believe that these considerations have important implications in the design of programs for problem solving applications.

1) *Scientific communities are structured to support competition as well as cooperation.* A system must be able to support parts of itself working on related or unrelated problems concurrently.
2) *Current adherence does not imply adherence for all future time.* Messages must be labeled with their context (author, time, place, etc.).
3) *Adherence is a local rather than a global phenomenon.* We would like to support plurality of approaches within the system. Different approaches naturally require somewhat different belief sets. There is no one who speaks for the scientific community as a whole.

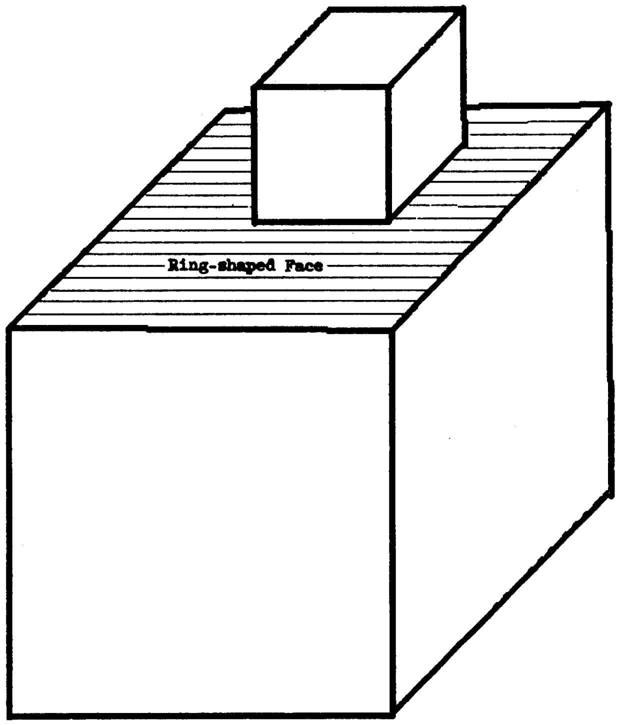

Fig. 1. Lakatos counterexample.

Often opposing directions of research are found between schools of thought that do not closely interact. An example is schools of psychology. The Freudian and behaviorist schools explain human neuroses in very different ways. Neither has proved that its ideas are in all respects superior to its competitors. There is no good reason for the community as a whole to accept one theory over the other. In the absence of criterial experiments to discredit one theory or the other, these opposing ideas may coexist in the community for a long time. Evolution is often gradual, taking years or decades. On rare occasions it is quite rapid as the community reacts to a "breakthrough" that clearly decides an issue previously muddled.

Research effort in opposing directions is not only found between distinct schools of thought; it can often be found in the same research group or individual. A tradition of intercolleague criticism of ideas is found in many research centers.

VIII. Viewpoints

The Ether examples of the previous sections did not involve any concept of *relativized belief*; in a sense, all assertions present were "believed" by the entire system. Relativized beliefs arise in accounting for the plurality of adherence in Ether. *Viewpoints* are a construct used in Ether to relativize messages as to assumptions, approaches, etc. From the sprites' point of view they represent *access points* to the messages in the system.

All thought happens in the context of certain assumptions. For example, we can distinguish between the *flat-*

world and *round-world* hypotheses by imagining two viewpoints, one for each of the theories. There are sprites that "watch for" messages which mention certain viewpoints. They may add new messages tagged with this viewpoint. We would expect statements in the flat-world viewpoint to the effect that the world has edges and the belief in the round-world viewpoint that traveling west long enough will bring you back to where you started.

A. Inheritance

Most information will be shared between viewpoints. For example knowledge about ships traveling on oceans will, for the most part, be the same in the round-world viewpoint as in the flat-world viewpoint. To make this methodology practical, an inheritance mechanism should be supplied for the messages in viewpoints. Both the flat-world and round-world viewpoints will inherit most of their information from viewpoints indicating general knowledge about the world.

Ether can be used to provide a simple definition of inheritance between viewpoints. Suppose that a viewpoint $v2$ inherits from viewpoint $v1$. Whenever there is a goal g with viewpoint $v2$ then it can be satisfied by attempting to show g with viewpoint $v1$. If g can be achieved with viewpoint $v1$ then it can be achieved with viewpoint $v2$. The following sprite shows how this procedure can be expressed very simply in Ether:

```
(WHEN (ASSERTION (INHERITSFROM =v2 =v1))
  (ACTIVATE (WHEN (GOAL =g (WITH VIEWPOINT v2))
    (SHOW (g (WITH VIEWPOINT v1))
      (WHENACHIEVED
        (ASSERT g (WITH VIEWPOINT v2))))))).
```

In effect the above sprite gives viewpoint $v2$ a "virtual copy" of the information in $v1$.

We can illustrate the use of inheritance by showing how to implement conditional proof in logic. Suppose the goal is to prove that a formula p implies a formula q in some viewpoint v. This goal can be achieved by creating a new viewpoint v' which inherits from v, assuming p in v', and showing that q holds in v'. The following sprite expresses exactly this method:

```
(WHEN (GOAL (IMPLIES p q) (WITH VIEWPOINT =v))
(LET (v' BE (CREATENEWVIEWPOINT (WITH PARENT v)))
      (ASSERT (INHERITSFROM v' v))
      (ASSERT p (WITH VIEWPOINT v'))
      (SHOW q (WITH VIEWPOINT v')
        (WHENACHIEVED
          (ASSERT (IMPLIES p q) (WITH VIEWPOINT v)))))).
```

People rarely consider any belief totally false. Instead, it may be advisable to accept the utility of a proposition *with certain provisos*. Newtonian mechanics, although not "correct" is still quite useful. Newtonian mechanics is now believed to be valid asymptotically for small velocities. In fact many engineering disciplines find "relativistic effects" insignificant enough that they can be safely ignored. This kind of inheritance might be expressed schematically in Ether as follows:

```
(WHEN {(GOAL (FORMULA = E
  (WITH RELATIVEVELOCITY = v))
    (WITH VIEWPOINT RELATIVISTIC))
      (ASSERTION (= v {APPROXIMATELY} 0))}
      (SHOW (FORMULA E)
        (WITH VIEWPOINT NEWTONIAN)
          (WHENACHIEVED
            (ASSERT (FORMULA E)
              (WITH VIEWPOINT RELATIVISTIC))))).
```

B. Translation

In many cases viewpoints although closely related do not inherit all information from one another. For example Newtonian mechanics is a special case of relativistic mechanics. However the frame transformations of Newtonian mechanics are not inherited from special relativity. The Newtonian frame transformations are *translations* of the relativistic ones. The following sprite translates all the relativistic transformations into their Newtonian counterparts:

```
(WHEN (ASSERTION (TRANSFORMATION = E (WITH RELA-
TIVEVELOCITY = v)) (WITH VIEWPOINT RELATIVISTIC))
    (ASSERT (TRANSFORMATION (LIMIT_{v→0} E)) (WITH VIEW-
POINT NEWTONIAN))).
```

Note that the above sprite does all the translation work in parallel and records all the results in the Newtonian point of view. This example differs from the previous ones illustrating inheritance in that all the translations are done on the grounds that they will prove to be generally useful instead of waiting for a problem in Newtonian mechanics that seems to need a transformation of a certain form. The style of reasoning illustrated above is often called "antecedent reasoning" or "reasoning forward from the data."

IX. FUTURE WORK

We have found the scientific community metaphor a useful one for stimulating the development of programing structures for artificial intelligence research. We plan to continue exploring this metaphor. We are now involved in the study of several application areas to be programmed in Ether using ideas developed in this paper. One of us (Kornfeld) is currently involved in research applying this problem solving approach to certain problems in program understanding and synthesis.

Pattern-directed invocation languages (such as Ether) pay dearly for their generality through lack of efficiency. We are now investigating a scheme known as *virtual collection of assertions* that we believe will make this class of system practical for much larger applications. Pattern-directed invocation languages have traditionally been implemented using methods that index messages in a uniform way. Sprites search this database for messages on which they can trigger. There are various schemes, such as discrimination nets and hash tables, that keep the speed of this search process from degenerating linearly with the number of messages. Knowledge can be stored and retrieved more readily when it is indexed in some *semantically* meaningful way. We propose to selectively augment the default indexing and retrieval methods. The new procedures say in effect "When you want to disseminate messages of a certain kind, what you *really* do is execute this procedure." The procedure will record the messages in a way that it can be efficiently retrieved. Correspondingly sprites which can trigger on messages or certain kinds perform compatible procedures. In this way we can gain the benefits of pattern-directed invocation without incurring all of the overhead.

The description system *Omega* [7] addresses issues involved in the construction and manipulation of semantic networks of structured descriptions. A description language has potential advantages over simple assertions as the medium for communication in Ether because semantic relationships we may want to talk about have already been encoded in the description language. The possibility exists for using Omega as the language by which communication in Ether happens. The efficient implementation of Ether involves the development of a solid basis in concepts of parallel communication. We hope to draw on the notions of message passing and serialization as developed in *Act 1* [9].

Although we have been talking about parallel *languages* we have not concerned ourselves with parallel *hardware*. We believe advantages of parallel processing languages for problem solving are substantial on a serial computer run in a time-sliced fashion. However, because of the inherent parallelism in Ether, parallel hardware could be effectively utilized. The *Apiary* [10], [21] is an architecture under construction at the Massachusetts Institute of Technology Artificial Intelligence Laboratory for implementing computer systems with large amounts of parallelism.

X. Conclusion

Our main point is that *scientific communities are highly parallel systems*. Many scientists are able to work concurrently on the same, similar, or quite different problems. At any one time, in a scientific community, alternative theories are expounded concurrently and argued. This diversity, rather than being superfluous to the growth of science, is essential to it.

We believe that these observations can supply important insights that will aid in the construction of problem solving systems. The Ether language has been designed to facilitate the construction of problem solvers with a high degree of concurrency and diversity. Ether programs do not begin to approach the complexity found in scientific communities; nevertheless, we believe the metaphor will prove fruitful in guiding the designs of future problem solvers.

XI. Related Work

Most of the ideas in the Ether system are not completely new. They are further developments of ideas in previous systems. This paper represents a further development of the scientific community metaphor for problem solving which was proposed as an important topic for investigation in [6]. The motivation is that incorporating the problem solving mechanisms of scientific communities can be fruitful in the same way that incorporating the problem solving mechanisms of individual humans.

The *parallel pattern directed invocation* used in Ether is a descendant of a pattern-directed invocation developed for Planner-69 [4] which was implemented under the name Micro-Planner. In Planner when a goal was proposed an applicable *demon* was chosen to attempt to achieve the goal. If the *demon* concluded that it had failed then it would backtrack and another applicable demon would be chosen. The *parallel* pattern directed invocation of Ether differs in that all the applicable sprites which can get support will work on the goal concurrently. In addition there is the possibility of even more parallelism because there may be some skeptics which would like to demonstrate that the goal cannot be achieved. In view of these differences, Danny Hillis suggested the name *sprite* to replace the somewhat malevolent *demons* of Planner. The nested continuation control structure of Ether builds on similar constructs in [23], [5], [1].

Viewpoints are a means of relativizing messages according to different assumptions, authors, times, and theories. The intellectual roots for viewpoints come from several sources. McCarthy introduced the notion of a situation argument in predicates to relativize collections according to a situation where a situation was defined to be the entire state of the universe. In his system each situation represented the entire state of the universe at one time. In the development of the actor theory of computation, this notion of global state proved to be ineffective so a theory of local states was developed in its stead. In QA-4 Rulifson introduced a context mechanism to Planner-like systems. Contexts could be used to represent hypothetical assumptions. The QA-4 notion of context was limited by being tied to a control structure of "pushing" and "popping" contexts and by the way in which a particular notion of inheritance was wired into the interpreter. Providing inheritance and translation sprites provides a more powerful and useful mechanism for addressing issues of sharing knowledge between viewpoints. The inheritance in the natural deduction system used in Ether closely resembles the logical system developed by Kalish and Montague [24].

Turing was one of the first to develop the notion of an individual human metaphor for problem solving through the notion of his famous Turing Test. This paradigm was developed and deepened with the development of Artificial Intelligence as an identifiable field of research in the 1950's. Newell and Simon did notable work within this paradigm through their work in analyzing the protocols of individual human problem solving behavior on puzzles. More recently Minsky and Papert have attempted to develop the idea that the mind of an individual human is composed of a society of agents. The above research on *individual* human problem solving is complementary to our research on the scientific community metaphor.

It may well be the case that some of the problem solving mechanisms which we have explored in the context of scientific communities are also used by the human brain. Unfortunately the current state of the art in neurophysiology is too primitive to shed much light on the question. One of the most important benefits of studying scientific communities is that we have been able to find convincing examples of the problem solving techniques discussed in this paper. Moreover investigation of the structure of problem solving in scientific communities has greatly helped in the design and evolution of Ether.

ACKNOWLEDGMENT

The authors would like to thank Randy Davis, Fry, Ken Forbus, Ken Kahn, David Levitt, and Henry Lieberman for reading earlier drafts of this paper and supplying helpful comments. Conversations with Roger Duffey, Harold Goldberger, Pat Hays, Nils Nilsson, Allen Newell, John McCarthy, Jeff Rulifson, Luc Steels, Gerry Sussman, Sten-Ake Tarnlund, and Richard Weyhrauch have proved very useful in developing these ideas.

REFERENCES

[1] J. de Kleer, J. Doyle, C. Rich, G. Steele, and G. Sussman, "AMORD: A deductive procedure system," Massachusetts Inst. Technol., Cambridge, MA, Artificial Intelligence Lab., Memo 435, Jan 1978.

[2] S. Fahlman, "A system for representing and using real-world knowledge," Ph.D. dissertation, Dep. Elec. Eng. and Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, 1978.

[3] P. Feyerabend, "Against method," Minnesota Studies in Phil. of Sci., 1970.

[4] C. Hewitt, "PLANNER: A language for manipulating models and proving theorems in a robot," presented at the First Int. Joint Conf. on Artificial Intelligence, 1969, Washington D.C., Aug. 1969.

[5] _____, "How to use what you know," presented at the Fourth Int. Joint Conf. on Artificial Intelligence, Tblisi, U.S.S.R., Sept. 1975.

[6] _____, "Viewing control structures as patterns of passing messages," *Artificial Intelligence J.*, vol. 8, pp. 323–364, June 1977.

[7] C. Hewitt, G. Attardi, and M. Simi, "Knowledge embedding in the description system OMEGA," in *Proc. AAAI*, Stanford, CA, Aug. 1980.

[8] C. Hewitt, G. Attardi, and H. Lieberman, "Specifying and proving properties of guardians for distributed systems," in *Semantics of Concurrent Computations* G. Kahn, Ed., *Lecture Notes in Computer Science*, No. 70. Berlin: Springer, 1976.

[9] _____, "Security and modularity in message passing," presented at the First Int. Conf. on Distributed Computing Systems, Huntsville, AL, Oct. 1979.

[10] C. Hewitt, "The Apiary network architecture for knowledgeable systems," in *Proc. Lisp Conf.*, Stanford, CA, Aug. 1980.

[11] W. Kornfeld, "Using parallel processing for problem solving," Massachusetts Inst. Technol., Cambridge, MA, Artificial Intelligence Lab., Memo 561, 1979.

[12] T. S. Kuhn, *The Structure of Scientific Revolutions*, 2nd ed. Chicago, IL: University of Chicago, 1970.

[13] I. Lakatos, *Proofs and Refutations*. New York: Cambridge University, 1976.

[14] _____, "Falsification and the methodology of scientific research programmes," in *Criticism and the Growth of Knowledge*, Musgrave and Lakatos, Eds. New York: Cambridge University, 1970.

[15] V. R. Lesser, L. D. Erman, "A retrospective view of the Hearsay II architecture," presented at the Fifth Int. Joint Conf. on Artificial Intelligence, 1977.

[16] J. McCarthy, "Situations, actions, and causal laws," Stanford University, Stanford, CA, Stanford Artificial Intelligence Project, Memo 2, 1963.

[17] J. McCarthy and P. Hays, "Some philosophical problems from the standpoint of artificial intelligence," Stanford University, Stanford, CA, Stanford Artificial Intelligence Proj., Memo No. AI-73, Nov. 1968.

[18] K. R. Popper, *Conjectures and Refutations*. New York: Basic Books, 1962.

[19] _____, *The Logic of Scientific Discovery*. New York, Harper & Row, 1968.

[20] J. F. Rulifson, J. A. Derksen, R. J. Waldinger, "QA4: A procedural calculus for intuitive reasoning," Stanford University, Stanford, CA, Stanford Research Inst. Artificial Intelligence Center, Tech. Note 73.

[21] J. Schiller, "The design and implementation of APIARY-0," 1979.

[22] R. G. Smith and R. Davis, "Distributed problem solving: The contract net approach," in *Proc. Second National Conf. of Canadian Soc. for Computational Studies of Intelligence*, July 1978.

[23] M. B. Wilber, "A QLISP reference manual," Stanford University, Stanford, CA, Stanford Research Inst. Artificial Intelligence Center, Tech. Note 118, 1975.

[24] D. Kalish and R. Montague, *Logic: Techniques of Formal Reasoning*. New York: Harcourt Brace Jovanovich, 1964.