Massachusetts Institute of Technology
Artificial Intelligence Laboratory

AIM 475

May 1978

Understanding in Incomplete Worlds

by

Steven Rosenberg

Most real world domains differ from the micro-worlds traditionally used in A. I. in that they have an incomplete factual database which changes over time. Understanding in these domains can be thought of as the generation of plausible inferences which are able to use the facts available, and respond to changes in them. A traditional rule interpreter such as Planner can be extended to construct plausible inferences in these domains by A) allowing assumptions to be made in applying rules, resulting in simplifications of rules which can be used in an incomplete database; B) monitoring the antecedents and consequents of a rule so that inferences can be maintained over a changing database. The resulting chains of inference can provide a dynamic description of an event. This allows general reasoning processes to be used to understand in domains for which large numbers of Schema-like templates have been proposed as the best model.

This report describes research done at the Artificial Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract Nooo14-75-C-0643.

Suppose that you were a farmer, and each week you must make decisions about your wheat crop based on the the price you expect to get for it. You formulate the question to yourself: "Will the price of wheat rise?" In making this decision you have available a stream of information ranging from weekly reports on global demand to your own knowledge about the weather and state of your crop. As an aid in making your decision you would like to know whenever these facts can be organized into a descriptive scenario which supports the hypothesis of rising wheat prices. (i.e. The question about the price of wheat is treated as a hypothesis about the domain.)

This is a task which involves "understanding". It is a type of task which can be difficult to do when the amount of information available is large and the information changes frequently. For instance, if you were charged with preparing the President's daily news summary, you would have to read hundreds of reports each morning, and analyze them in sufficient detail to summarize only those parts which were relevant to the president's concerns.

Understanding of this sort is a form of problem solving. It has two distinguishing features. First, it is reasoning in the service of perception. That is, it is particularly concerned with how related information from a variety of sources is organized into data structures which can in turn serve as input to further reasoning processes. This sort of fairly automatic organization of semantic memory is something people do without much trouble. This suggests the need for a corresponding set of computational procedures which can handle this task in intelligent support systems. Secondly, most real world databases share certain features of incompleteness and instability which make traditional reasoning processes break down. Once again, special mechanisms are needed to ensure that

inferencing can proceed correctly.

In this paper I will discuss the design of a program, Sleuth, for performing this type of task. Sleuth's purpose is to recognize events in real world domains by making inferences which support the questions it is given, and to keep these scenarios current over a changing database. Sleuth is not itself a theory of problem solving strategy, since it does not involve a commitment to a particular approach such as means-end analysis (Ernst and Newell, 1969) or procedural nets (Sacerdoti, 1977). Rather, Sleuth is an approach to the interpretation and construction of reasoning rules which allows them to be used successfully in certain types of real world domains.

Real world domains often don't seem to demand great problem solving efforts in order for most people to operate reasonably well in them. Many of these domains share certain features. They are

A) Incomplete

B) Unstable

They are incomplete because not all the information we might need in order to make an inference is available at a particular time. For instance, you may have to decide if the price of wheat is going to rise, when you know only the supply, but not the demand for wheat.

They are unstable since the particular subset of information available can change fairly rapidly in the real world. For example, a farmer receives daily weather reports, weekly crop surveys, daily market prices, and so on.

Knowledge based approaches to understanding in common sense domains have focused on the use of frames or scripts (Cullingford, 1977; DeJong, 1977; Reiger, 1978). This has been a successful approach in domains with well structured semantics in which knowledge is added in a highly organized fashion, such as stories and newspaper articles. In such cases it has been

profitable to view the new knowledge as instantiating a small set of static frames or scripts. However, for complex events it is very difficult to provide enough scripts or frames to encompass all alternatives. Relevent information may occur in several different stories or reports. These may be added at different times.

Sleuth is intended as a model of what it means to "understand" knowledge in some real world domains. Like all such models, it focuses on a particular subset of issues and problems and ignores others. Sleuth is an attempt to provide a more unified view of problem solving and understanding by considering the creation of event descriptions from a given set of assertions to involve a series of inferences which link these assertions in support of some central hypothesis. This approach ignores issues which more knowledge based theories have focused on, such as the use of prior knowledge of the domain (Charniak, 1977), or the usefulness of powerful sets of semantic primitives in making plans (Wilensky, 1976). McDonald (1978) has reviewed these three approaches to story understanding in terms of the strengths and weaknesses of each approach. Ultimately, a complete theory of understanding will have to incorporate elements of many of these partial models.

In contrast to these knowledge-based approaches I will argue that in real world domains where the knowledge is not necessarily pre-structured into simple stories, an understanding system will have to have capabilities that can best be characterized as extensions to traditional approaches to inferencing. Like Charniak and Wilensky, I have focused on the particular type of task which most demands just those features which I wish to emphasize. Complex events in real world domains have properties which suggest that a dynamic approach, drawing on inferencing strategies, may be the best solution.

Events have fuzzy boundaries: Consider a complex event, such as the recent Israeli invasion of Lebanon. What are the components of this event? We could define it as consisting of all armed clashes between Israeli forces and Palestinians after Israeli troops crossed the border. However, some people might also include the previous raid into Israel which provoked the invasion. We could go even further, and expand our description to include the United Nations meetings on the invasion, the attitude of Syria, or even the events that led to the creation of the Palestinian refugees. There is no fixed definition or single frame for this event. Like many others it has fuzzy boundaries. Using inferences to link the available facts yields a dynamic description for an event. The "boundaries" can be extended at will by making further inferences to include more facts.

Events are not "things": For example, consider Israeli versus Palestinian descriptions of the invasion. These may not include the same set of facts. When the two descriptions do share facts in common, they will be organized by different relations. Inference processes are flexible enough to capture this distinction by applying different rules of inference to some initial hypothesis to generate different chains of inference linking different (or partially different) sets of assertions in support of the same central hypothesis.

Events have variable instantiations: For any particular type of event, a different subset of features may be missing for each instance. By using chains of inference to connect assertions we can evaluate the plausibility of any particular event description without necessarily having to specify beforehand all acceptable partial instantiations.

Anything could be relevant: If we ignore plausibility, we can create an event description with almost any set of assertions. For example, if we follow sports we might postulate a scenario in which the Texas Rangers win

the world series. We do not want to exclude such far fetched scenarios a priori from the range of possibilities. Indeed, we wish to do just the opposite. We would like to retain the ability to create event descriptions which involve a given set of facts, and then judge whether the scenario is plausible or not. This is useful in hypothetical reasoning, and allows us to adjust the false alarm rate to correspond to the expected utility of the result.

Special problems arise in recognizing an event due to its essentially dynamic nature. For instance the assertions comprising a particular event may be added to the database over a period of time. This corresponds to fact that events occur sequentially over time. Sequential instantiation of an event can lead to difficulties. Current assertions may become obsolete, change or be deleted before the final components of the description are added to the database. An event description is built on shifting sands, so to speak, and in generating these descriptions we must be able to respond to these changes. The recognition process involves conditionality. Where we go depends on what has gone before. For example, in a condition called Frost Heaving, a sequence of thaws and subsequent freezes can tear the roots of the winter wheat crop. If however, a thaw continues for more than a short time, the ground will thaw down to the base of the plant roots. The next freeze will then not tear the roots as the ground refreezes. Consequently, the duration of the freezes and thaws is as important as the number. Somehow we must keep a history of the calculation. This will allow us to determine which features to attend to based on the features we have already seen.

INCOMPLETE DOMAINS

In this section I will discuss how a traditional reasoning program, such as Planner, can be extended for use in a domain with incomplete information. Consider, for example, how a farmer, (let's call him Farmer MacDonald), might go about making weekly decisions concerning his crop of wheat. He formulates the question: "Will the price of wheat rise?". If he can generate a scenario from available information which supports the notion of the price rising, he will adopt one set of farming strategies; if not, another. If Farmer Macdonald has a consultant program whose reasoning processes resembled Planner (Sussman, Winograd, and Charniak, 1971), he might formulate the problem as follows:

```
(Thgoal (price-increase wheat))
```

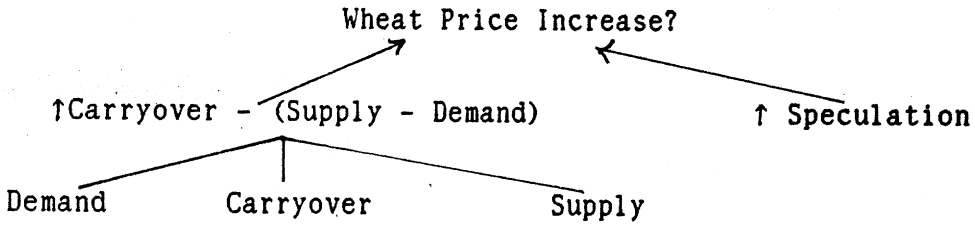
This goal is a hypothesis about the state of the world. The consultant makes inferences on the current set of assertions to see if an event description supporting it can be generated. We can define an event as being a set of assertions, justified by rule instances, which support some central hypothesis. If farmer MacDonald's assistant knows the following theorems and assertions, it will be able to construct a plausible chain of inference:

```
(Thassert Supply Wheat 180,000,000 bushels)
(Thassert Demand Wheat 170,000,000 bushels)
(Thassert Carryover Wheat 15,000,000 bushels)
(Thassert old-supply wheat 182,000,000 bushels)

(Thconse Thm1 (X) (price-increase ?X)
     (Thor (Thgoal (Supply-&-demand ?X))
           (Thgoal (Speculation ?X)))

(Thconse Thm2 (X S D C) (Supply-&-Demand ?X)
     (Thcond ((Thand (Thgoal (supply ?X ?S bushels))
                     (Thgoal (demand ?X ?D bushels))
                     (Thgoal (carryover ?X ?C bushels))
                (greaterp (?C) (- ?S ?D))))
```

Thm1 specifies that to show a price increase for wheat, either of two subgoals can be attempted. The first subgoal specifies a pattern which matches that of Thm2. Thm2 will succeed only if the difference between supply and demand is less this year than last. If so, Thm1 will succeed. A price increase for wheat is inferred, and Farmer Macdonald can increase his wheat plantings. (Note that we are simplifying the decision processes of a farmer; he has not looked at demand for alternate crops; whether his production costs on wheat have gone up; nor what sort of growing season is predicted. However, our purpose is not to show how Planner can be used in farming, but to illustrate some limitations of Planner.) We can illustrate this graphicly as follows:



Current demand for wheat is reported weekly by the U.S.D.A., based on domestic reports, and satellite observations of foreign lands. Each week, as farmer Macdonald reads his newsletter, he marks existing assertions as old, and adds new ones.

e.g. Current information is marked as old, or erased:

```
(Thgoal (demand wheat ?D bushels))
(Thassert (old-demand wheat ?D bushels))
(Therase (demand wheat ?D bushels))
```

New information is asserted:

```
(Thassert (demand wheat 180,000,000 bushels)
```

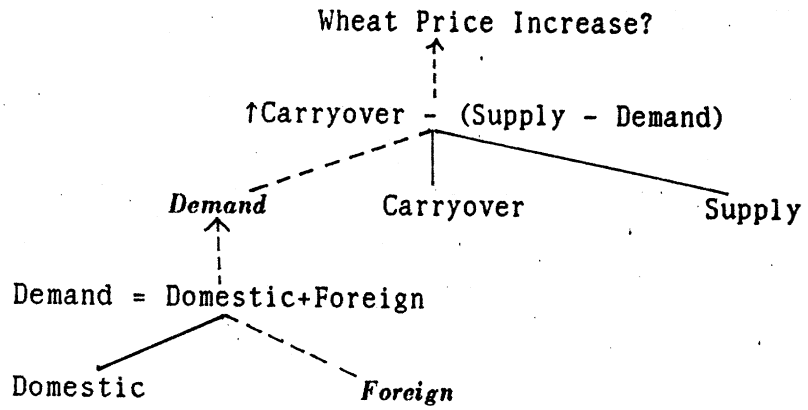
Suppose that the next week, due to very foggy weather in central Asia, no satellite photos are taken. As a result, the U.S.D.A. issues no new

demand statistics. This time, when the consultant is asked about the price of wheat, the previous inferences would fail, since no assertion matching the pattern for (Demand Wheat ?D bushels) would be found. For such cases, Planner provides a strategy. If an assertion is not in the database, Planner will try and prove it:

```
(Thconse Thm3 (X DD FD) (demand ?X)
  (Thcond ((Thand (thgoal (domestic-demand ?DD))
                  (thgoal (foreign-demand ?FD)))
           (Plus ?DD ?FD))))
```

Thm3 states that to deduce demand for a commodity, find the foreign and domestic demand for this commodity, and add these together. If there are assertions for foreign and domestic demand, this theorem would succeed. However, since the total demand reported by the U.S.D.A. is based on the missing estimates of foreign demand, Thm3 will also fail. In this case, if there are no other methods for proving the missing assertion, the consultant must give up.

(In this and subsequent diagrams italics are used to indicate antecedents which are missing. A solid line is used to connect antecedents to rules; similarly, a solid line and arrow is used to indicate deduced links between rules and assertions, while a dashed line and arrow indicates inferences which have failed.)



This presents a fairly brittle mechanism for dealing with domains which share the properties of incompleteness and instability. If the needed

assertions are not in the database, and if they cannot be infered, the inference attempt will fail. It will fail, however, not necessarily because it is wrong, but because not enough information exists to make inferences. People are not quite so brittle reasoners, since they often cannot postpone decisions until more knowledge is available. For example, Farmer MacDonald might reason that as long as the supply of wheat is decreasing, it will be worthwhile to plant more regardless of demand. (Note: I am not suggesting that this is the "best" decision; merely that it is a plausible decision, and represents the sort of flexible reasoning which people are capable of.) By being willing to make assumptions, Farmer MacDonald is able to use a form of a rule which can operate on the information available. We might express this by adding a second clause to the Thcond of Thm2:

```
(Thconse Thm2A (X S D C) (Supply-&-demand ?X)
  (Thcond ((Thand (Thgoal (supply ?X ?S bushels))
                  (Thgoal (demand ?X ?D bushels))
                  (Thgoal (carryover ?x ?C bushels)))
           (greaterp ?C (- ?S ?D)))
          (Thor (Thgoal (supply-decrease ?X))
                (Thgoal (demand-increase ?X)))))
```

If the full set of assertions concerning supply, demand and carryover are unavailable, Thm2A now suggests either trying to prove that supply has decreased, or demand has increased. By creating goals that require only a subset of the assertions that the original theorem required, Thm2A starts to capture the notion of rule simplification.

However, Thm2A does not quite capture our intuitions about simplifications. A rule should give advice about which other theorems can function in its stead as simplifications. We can then choose to use this advice or not, depending on our strategy. By treating simplifications like other goals in Thm2A, we lose this intuition. More importantly, while we can express simplifications of rules as theorems of the same sort as other theorems, they are not equivalent to other consequent theorems. A

simplification is not used as a sub-goal in instantiating a theorem. Instead it replaces that theorem and is an alternative method for proving that theorem's goals *given the conditional circumstances of the theorem failing, and our willingness to make assumptions.* The simplifications, for the purposes of making inferences, are considered equivalent to the original theorem under the given circumstances, although ordinarily the simplifications achieve different goals than the theorem they replace.

A rule can have more than one procedural counterpart. Part of Planner's contribution to the notion of pattern directed invocation of rules was the insight that a rule has both a consequent and antecedent meaning. These can be expressed as two classes of theorems, antecedent and consequent theorems, which can be invoked by different patterns. We can extend this a step further by postulating that a rule has another bundle of procedural counterparts corresponding to its simplifications.

Actually, these simplifications will simply be other rules. However the knowledge of when these other rules can be used as simplifications, and which rules can be used, must be represented. Sleuth can be viewed as an intelligent interpreter which can make use of this metaknowledge (Davis & Buchanan, 1977) to substitute simpler theorems for a rule which fails. The appropriate place to specify this metaknowledge is in a separate class of theorems. e.g.

```
(Thconse Thm2 (X S D C) (supply-&-demand ?X)
     (Thcond ((Thand (Thgoal (supply ?X ?S bushels))
                     (Thgoal (demand ?X ?D bushels))
                     (Thgoal (carryover ?X ?C bushels)))
              (greaterp ?C (- ?S ?D)))))

(Thassume Thm4A (X) (Supply-&-demand ?X)
     (Thgoal (supply-decrease ?X))
     (Thcaveat (Default)))

(Thassume Thm4B (X) (Supply-&-Demand ?X)
     (Thgoal (demand-increase ?X))
     (Thcaveat (Thnot (Thgoal (supply-increase ?X)))))
```

We now introduce a new class of theorems, such as Thm4A and B, indicated by the label Thassume. These theorems contain information concerning simplifications and assumptions. A Thassumption will specify A) a goal; theorems satisfying this goal can function as a simplification; B) the assumptions involved in using that simplification. These are expressed as a caveat. If the assumption being made is that the missing antecedents can be ignored, the caveat will contain a Default. If there is some particular condition which must obtain in order to assume that the antecedents can be ignored, this will be expressed in the caveat. (An alternative approach is to use a generative theory of simplifications in which a rule can be examined and a simplfication generated dynamically. The present solution can be thought of as the end step of such a process. The specification of the domain and metaknowledge necessary to achieve this is a complex task, which I have defered until the next iteration. However, see Carr and Goldstein (1977) for a model of how this metaknowledge looks in one domain.)

For instance, in Thm4A, proving a decrease in wheat supply can function as a simplification of Thm2 (proving a decrease in the difference between supply and demand), and hence can in this case be used to prove the goal of an increased price for wheat. However, ordinarily Thm4A would never match the goal pattern which invokes Thm2. Since no assumptions are specified in the Caveat, these can be ignored. This is explicitly expressed in the caveat as a default. If instead we use the goal of an increase in demand as a simplification, as in Thm4B, we must take account of the caveat that supply must not have increased for this simplification to be valid.

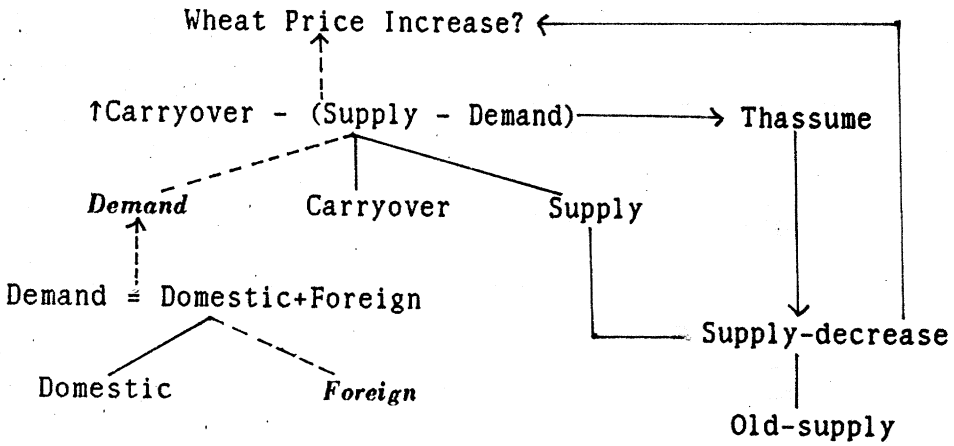
When Thm2 fails, Sleuth can choose to make assumptions which will allow a simplification to succeed on the assertions which are available, by using Thm4A or B. A rule and its associated simplifications are related through

the set of assumptions they embody. When Farmer MacDonald decides to ignore the demand for wheat, he is doing so because he is willing to assume that if demand changes, it will not change in a direction which would invalidate his reasoning. By making explicit this notion of assumptions, we can extend the list of options available in using a theorem to achieve a goal.

After Thm2 fails, simplifications will be considered, and Thm4A Found. Thm4A first tries to satisfy the goal (Thgoal (Supply-decrease ?X)). No assertion matching this pattern exists. However, a theorem, Thm5, can be used to prove this assertion.

```
(Thconse Thm5 (X S OS) (Supply-decrease ?X)
   (Thcond ((Thand (Thgoal (Supply ?X ?S bushels))
                   (Thgoal (old-supply ?X ?OS bushels)))
            (Thcond ((greaterp ?OS ?S))))))
```

The value for old-supply was one of the initial set of four assertions. Since current supply and old-supply are known, Thm5 will succeed, and support the hypothesis of higher wheat prices.



This chain of inferences results in a less plausible scenario than one requiring no simplifications.

UNSTABLE DOMAINS

The outcome of an attempt to understand some set of data by supporting a hypothesis is a set of assertions and rule instances, joined by various labeled links. This is a process trace. It is more than just a trace of a proof since failed rules and unsuccessful proofs are also recorded. The annotation available at any given time represent the "understanding" of the domain. As the database changes, Sleuth will try and maintain its hypotheses. This will be reflected in the changing set of annotation associated with each hypothesis. Sleuth assumes support for an hypothesis is conditional on the assertions available at the time it was first considered. Hence this support must be monitored and changed as the database changes. All proofs are conditional on the validity of the assertions used by the rules in the proof. However in many domains, once deductions are made from a set of assertions, no effort is made to insure that while the results of those deductions are used, the assertions still hold true. Generally, the user does not expect the database to change so as to invalidate prior inferences.

Sleuth extends the concept of rule interpretation by making the maintenance of goals a function of the interpretation of rules. Sleuth does this by giving each active rule the autonomy to respond to changes in its environment. As each rule is interpreted, Sleuth creates an associated Sentinel for that instance of the rule. The sentinel gives the rule instance the knowledge of how to respond to changes in its antecedents or consequents. The result is the maintenance of hypotheses through a method of local autonomy.

By using sentinels, Sleuth extends the basic idea of a rule which is evaluated successfully if its antecedents are satisfied at the initial time of evaluation. A rule instance must be continuously enabled while it is used in support of some hypothesis. Before describing sentinels we must

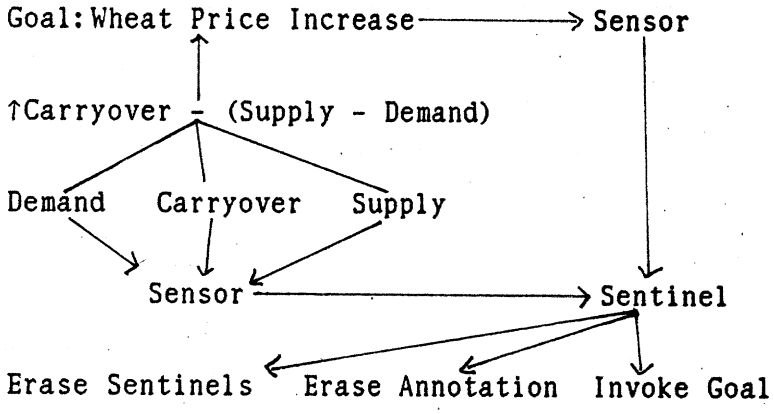
first consider under which conditions we wish the rule instance to be enabled.

1) An instance of a rule used in support of some hypothesis must continue to have its conditions met while that hypothesis is successful.

2) We can extend this to theorems which fail. These can also be monitored, as long as the hypothesis is successful. Failed rules may succeed at subsequent times, if the necessary antecedents are asserted or proved.

3) This notion can be extended one step further. The hypothesis may have initially failed. However, the goal of supporting that hypothesis is maintained. In that case, the theorems attempted are still active, although no event description or scenario exists. If, at a later time, they can succeed, they will reactivate the attempt.

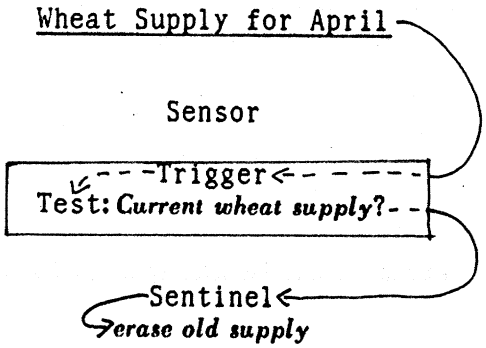
Sentinels were first developed by myself and Jim Stansfield as an aid in instantiating frames and automating the recognition of simple sequences of events within a changing database, using FRL (Roberts and Goldstein, 1977). Sleuth extends this by associating sentinels with the application of rules, and by making this use of sentinels a property of the rule interpreter. A sentinel associated with a rule instance will look like:



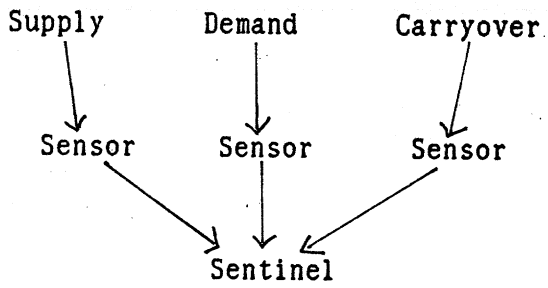
A triggered sentinel can take a variety of actions. The standard ones, indicated in the above diagram, are to A) erase itself; B) erase the

annotatation; C) reinvoke a goal.

A sentinel has sensors which report to it: (Sensor => Sentinel). A sensor has a two part condition. The first part, a trigger, is a demon which responds to changes in the pattern that triggers it. For example, in the following case, the trigger responds to any addition or deletion of patterns involving the wheat supply. The sensor then tests the pattern against some criterion. For instance, this sensor is only interested in assertions concerning current wheat supply:

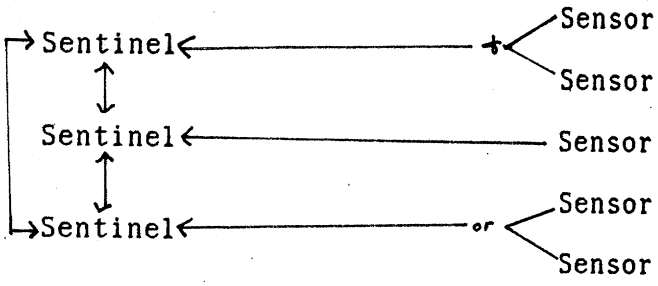


A sentinel can have several sensors which report to it. The sentinel is satisfied when some arbitrary logical conjunction of its sensors succeed. Although for the task of maintaining hypotheses more complex relations are not needed, a sentinel has the capacity to evaluate conditional relations among its sensors, and even to remove current sensors and place new ones as a response to these conditional constraints. It can also make use of the temporal dimension in conjunction with the logical organization of its sensors. For instance, an "and" relation among these sensors can be created in which all sensors are satisfied at the time the sentinel is evoked, or the relation among the sensors can be that they were all satisfied at some preceeding time (and if desired, in some specific order) but at the time the last sensor is satisfied, and evokes the sentinel, the state of the other sensors is unknown.

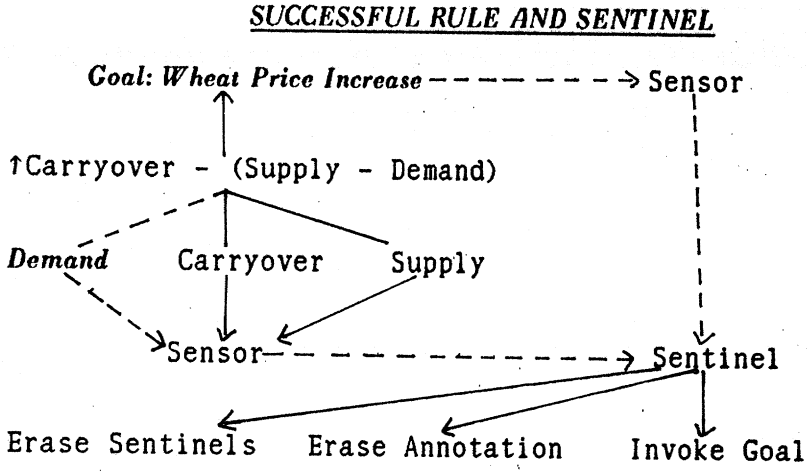


The sensors function as triggers for the sentinel, which is "data driven". A sentinel and its sensors are theorems which are created for a specific purpose. Unlike other theorems in the database, they have a limited lifespan. A sentinel can choose to erase itself and its sensors upon completing its goal. For the current task, sentinels are not required when their associated rule instance is no longer active. In this case, the sentinel will erase itself.

Since there are several aspects of a rule's environment we may wish monitored, we can create a cluster of sentinels, each of which is responsible for one aspect. Typically, we will want to be able to monitor such things as a rule's antecedents, sub-rules it requires for its own success, the goal which invoked the rule, and more powerful rules which could supercede a rule if they were used. Not all of these will be monitored at any one time. The interpreter which applies a rule will know which of these should be monitored. Should the sentinel looking at any one of these succeed, the result is often some action which renders the other associated sentinels obsolete. For instance, if the sentinel monitoring the goal is triggered by the erasure of this goal, its action will be to erase the instance of the rule it is associated with. As a result, the other sentinels are not needed. Consequently, the sentinels in a cluster have the ability to erase each other when the success of one removes the réason d'etre of the others.



Sleuth creates, for each instance of a rule, a cluster of sentinels which monitor the rule's antecedents, goals, and annotation while the rule is active. In the following diagrams individual clusters of sensors and sentinels have been collapsed into single sensors and sentinels, for illustrative purposes. Lets examine what happens with a successful rule when its sentinel is triggered.

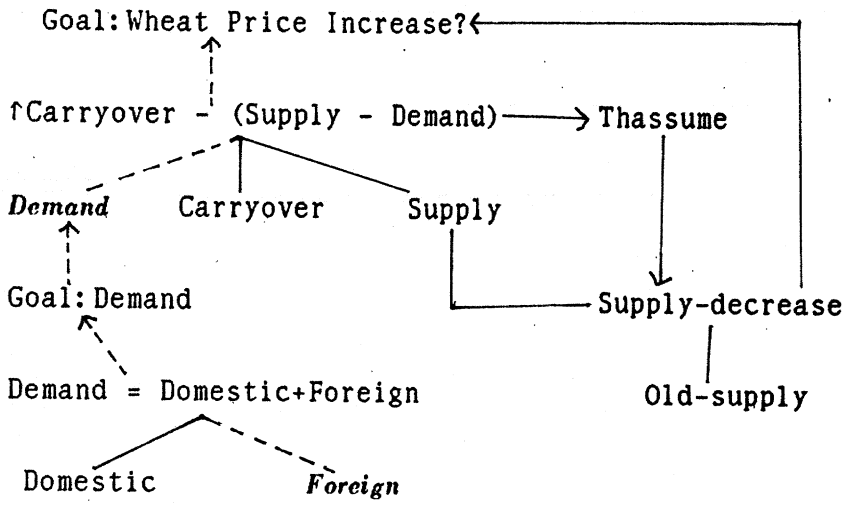


Individual rules will succeed or fail as a function of their antecedents. When a successful rule's antecedents change, its sentinel will be triggered. When this happens the sentinel causes the goal the rule was supporting to be re-evaluated. It removes the old annotation, as new annotation is created for the new evaluation of the goal. At this point the sentinel can erase itself. (Note: A sentinel causes a goal to be re-evaluated. There is no constraint that Sleuth must use the same rule again. However, in this and the following examples, it is assumed that there have

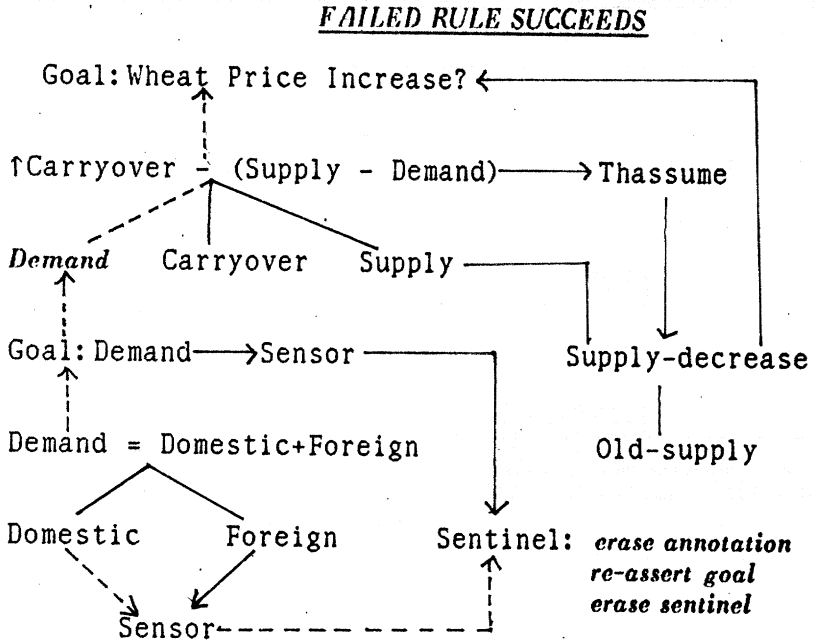
been no other changes in the state of the system that would cause another rule to be selected first.)

If the rule's goal changes, perhaps because we are no longer interested in the question it supports, the sentinel will also be triggered. In this case we do not wish to re-evaluate the goal. The sentinel will remove itself and erase the associated annotation. Thus the rule instance will no longer be active, since no trace of it will remain.

Now let's consider how this local association of rule instances with sentinels can give rise to the right global behavior. The following represents the state of our deduction so far:

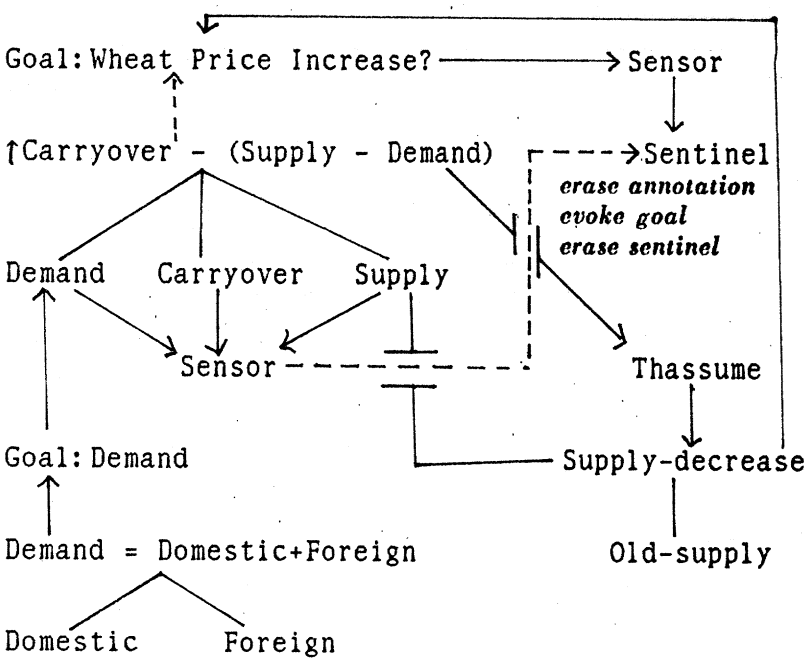


Suppose that a failed rule now is capable of succeeding, through its missing antecedent being asserted. For instance, the missing foreign demand for wheat can be asserted. This would trigger the sentinel associated with that rule instance:

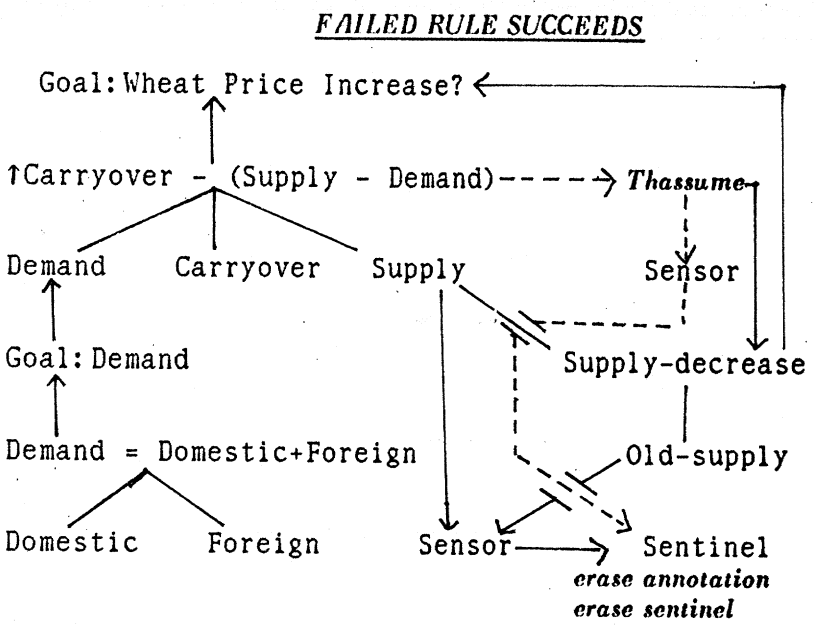


This will trigger the associated sentinel to erase the annotation for this rule instance, reassert the goal as something to be proved, and then to erase itself. This time the rule succeeds, resulting in a proof of the missing demand for wheat. This will in turn trigger the sentinel associated with the rule instance of Thm2 for which the missing demand is an antecedent:

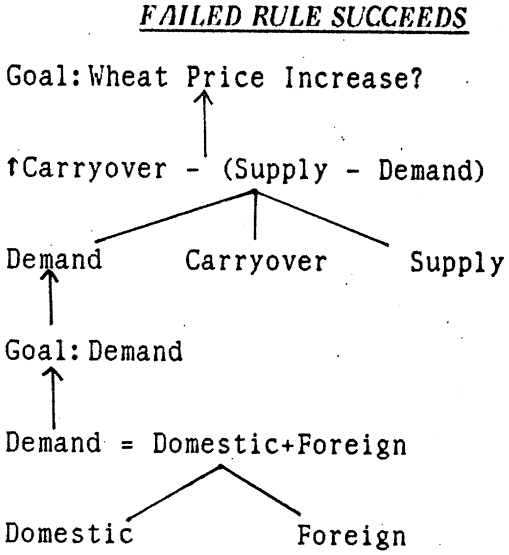
FAILED RULE SUCCEEDS



This sentinel repeats the actions of the prior sentinel. However, in erasing the annotation, it erases the record of the assumption made. This will trigger the sentinel on the rule which is a simplification. Since the use of a simplification is conditional on another rule failing, the sentinels associated with theorems used as simplifications monitor the annotation recording that failure, so that they will know when the simplification is no longer required. They will then respond to the erasure of this annotation by erasing the annotation for the simplification.

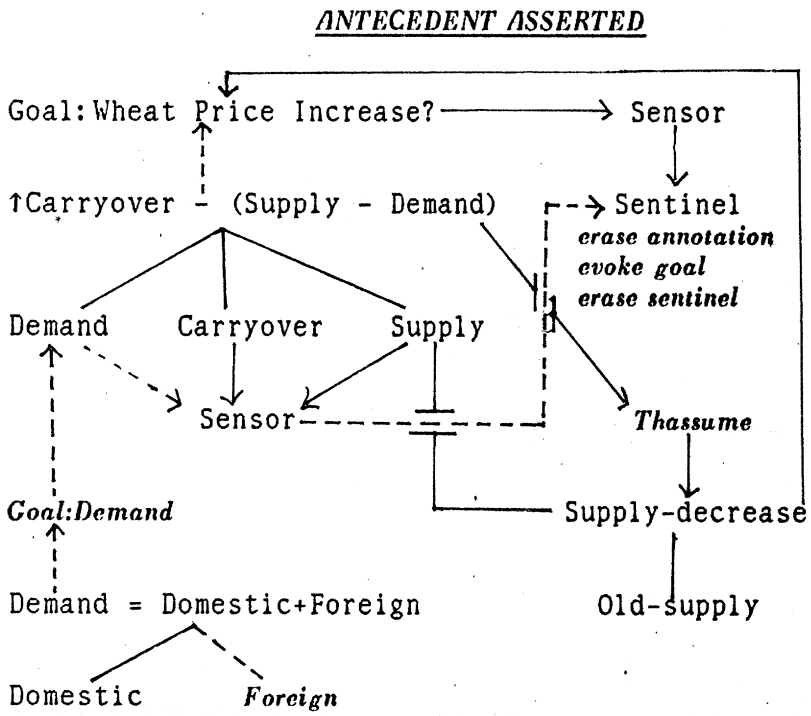


If Thm2 again failed due to a missing antecedent, Sleuth would once more try a simplification. Since the formerly missing antecedent for demand has now been infered, Thm2 succeeds, and results in the following final proof tree:

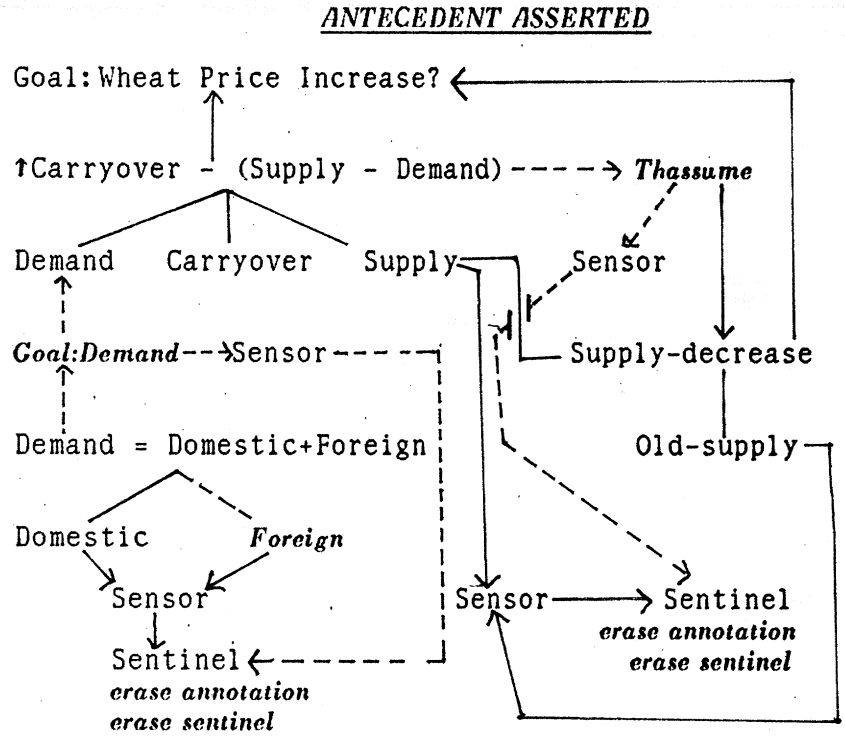


In this next example, the missing assertion for current wheat demand is asserted, although the rule involved (Thm2) has already "succeeded" by using a simplification. This will trigger the sentinel associated with the rule

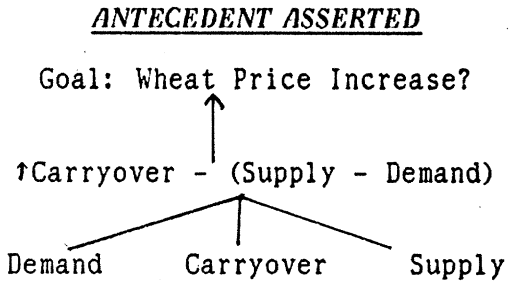
instance of Thm2:



When Thm2 is re-evaluated it succeeds without recourse to either using a simplification or trying to prove the now not missing antecedent. There is no explicit mechanism responsible for removing the now unneeded rule instances. Instead, by erasing its annotation, Thm2 triggers the sentinels associated with the subgoal of proving demand, and the simplification:



Both these sentinels, since their reason for existing is gone, erase the annotation, and then erase themselves. This results in the following final state:



Thus, unneeded rule instances will know when to remove themselves. Through local propagation, the representation responds to changes in the available database. Consequently, once a question has been specified, a dynamic process is invoked which once attempted can be locally data-driven. These changes will reinvoke the goal of inferencing, which can then proceed in a goal driven fashion. Obsolete parts of the representation are able to remove themselves by noticing local changes in the environment.

Sleuth, once given a goal, will attempt to recognize this event whenever the database contains the right set of assertions. Sentinels set in the interpretation of rules will individually call Sleuth to re-evaluate particular goals. Sleuth will develop new ways of supporting its hypotheses in response to these local calls for re-evaluation. Once applied, each sentinel has the autonomy to respond to changes in the database.

Sleuth is not a reasoning system per se. It is a set of features for an interpreter applying a reasoning strategy to some domain. These features are designed to allow reasoning to proceed over a changing database. Thus, standard reasoning techniques could be used in conjunction with Sleuth, if the domain warrented it. Perhaps the closest approach to Sleuth has been that of Doyle (Doyle, 1977). Doyle has provided a mechanism for recording deductive dependencies so that when facts turn out to be incorrect, the entire "context" of dependent facts can be removed from the database. This work develops the notion of dependency directed backtracking by making the use of contexts very explicit, just as Conniver (Sussman & McDermott, 1972) did with the chronological backtracking of Planner. Sleuth does not maintain deductive dependencies, since the changes it is designed to respond to are those imposed on the outside world by changes in the data, and not ones due to deduced inconsistencies. However, Sleuth does provide an automatic control structure so that the user can ignore the problems of a changing database, and focus on the deductions he wishes to make.

Sleuth is currently being programmed in FRL (Frame Representation Language), and not in Planner, which was chosen for the examples since it is a well understood language. While the ability to use simplifications and to use sentinels in monitoring rule instances exist in the current version, the Planner like features of chronological backtracking, and pattern-directed invocation of rules exist only in a rudimentary form.

The fact that Sleuth is currently designed to operate in a frame representation language suggests that this approach and the knowledge based approaches are not opposites, but instead are complementary. In FRL, Sleuth provides deduced links between frames which themselves have a structure, and which have been used in discourse understanding through the standard approach of frame instantiation (Sidner, 1978). Thus an inferential approach to understanding can take over where frame instantiation leaves off: for descriptions which occur infrequently enough so that no script exists, or which are so complex that there is no one cannonical characterization.

ACKNOWLEDGEMENTS

I would like to thank Ira Goldstein and Jim Stansfield for their helpful comments and advice.

REFERENCES

Carr, Brian & Goldstein, Ira P. 1977, "Overlays: A Theory for Computer Aided Instruction", MIT-AI Memo 406.

Charniak, Eugene. 1977, "A Framed PAINTING: the Representation of a Common Sense Knowledge Fragment", Cognitive Science, 1, 4.

Cullingford, Richard E. 1977, "Controlling Inference in Story Understanding", Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge, Mass.

Davis, Randall, & Buchanan, Bruce G. 1977, "Meta-Level Knowledge: Overview and Applications", Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge, Mass..

DeJong, Gerald. 1977, "Skimming Newspaper Stories By Computer", Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge, Mass.

Doyle, John. 1977, "Truth Maintenence Systems for Problem Solving", M.I.T. Artificial Intelligence Laboratory Technical Report 419.

Ernst, George W. & Newell, Allen. 1969, GPS: A Case Study In Generality And Problem Solving, Academic Press, New York, N.Y.

McDonald, David. 1978, "Story Understanding: the Beginning of a Concensus", MIT-AI Working Paper 168.

Reiger, Charles. "The Magic Grinder For Story Comprehension". To appear in Discourse Processes, A Multidisciplinary Journal.

Roberts, Bruce R. & Goldstein, Ira P. 1977, "The FRL Manual", MIT-AI Memo 409.

Sacerdoti, Earl D. A Structure for Plans and Behavior. 1977, Elsevier North-Holland, New York, N.Y.

Sidner, Candace. 1978, "A Progress Report on the Discourse and Reference Components of Pal", MIT-AI Memo 468.

Sussman, Gerald J., Winograd, Terry, & Charniak, Eugene. 1971, "Micro-Planner Reference Manual", MIT-AI Memo 203A.

Sussman, Gerald J, & McDermott, Drew Vincent. 1972, "Why Coniving is Better Than Planning", MIT-AI Memo 255A.

Wilensky, Robert. 1976, "Using Plans to Understand Natural Language", Proceedings of the ACM, Houston.