KNOWLEDGE REPRESENTATION IN AN EXPERT STORM FORECASTING SYSTEM

Renee Elio*
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1

Johannes de Haan
Computing Department
Alberta Research Council
Edmonton, Alberta T6G 2C2

ABSTRACT

METEOR is a rule- and frame-based system for short-term (3-18 hour) severe convective storm forecasting. This task requires a framework that supports inferences about the temporal and spatial features of meteorological changes. Initial predictions are based on interpretations of contour maps generated by statistical predictors of storm severity, Ib confirm these predictions, METEOR considers additional quantitative measurements, ongoing meteorological conditions and events, and how the expert forecaster interprets these extra factors. Meteorological events are derived from interpreting human observations of weather conditions in the forecast area. To accommodate the large amounts of different types of knowledge characterizing this problem, a number of extensions to the rule and frame representations were developed. These extensions include a view scheme to direct property inheritance through intermingled hierarchies and the automatic generation of production system rules from frame descriptions on an as-needed basis for event recognition.

I INTRODUCTION

The weather prediction task offers some particular challenges for expert system research and development. It is a domain characterized by complex scientific theories (e.g., thermodynamics), assorted models and statistical predictors, and huge quantities of meteorological data that have both spatial and temporal characteristics. Nonetheless, forecasting is still considered something of a "black art," if only because those models are not yet perfectly developed to allow forecasters to make consistently accurate, fine-grain predictions. With reality always providing the opportunity for perfect hindsight, a meteorologist can also acquire a repertoire of heuristics. Often, the art of expert forecasting resides in the expert's knowledge of how the statistical predictors or models are affected by idiosyncratic features of his geographical region. In this regard, the problem is similar to many applications that have routinely been approached with a combination of statistical methods plus "local" expertise.

We have been developing an expert system, called METEOR, that performs a specialised forecasting task—the short-term (3-16 hours) prediction of severe storms, usually hail storms. METEOR predicts where storms will initiate, how intense they will be, and where they will move. One of our design goals was to automate as much of this prediction task as possible. This means that METEOR starts with much of the same raw data that the expert uses—meteorological measurements, maps describing features on a large scale, and reports of conditions provided by trained observers at weather stations. This contrasts with an approach whereby the user characterizes the data or describes the problem in some symbolic form that an expert system can directly use (for instance, as elements to match rule conditions in a production system). METEOR first derives what the state of the world is and then applies the expert's strategy, knowledge, and heuristics to interpret this knowledge and make predictions.

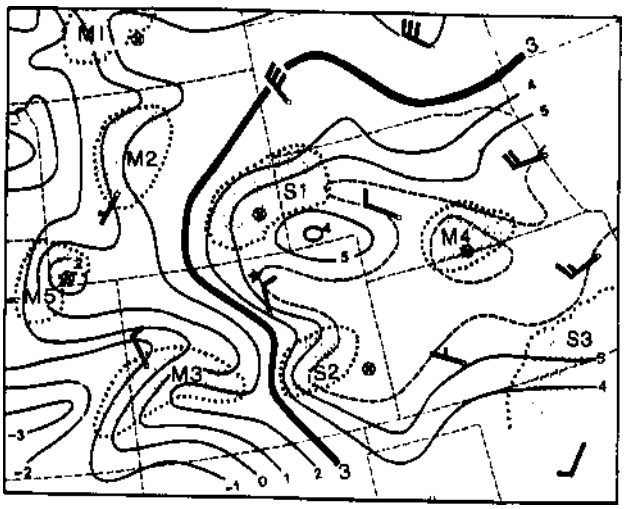
METEOR is not a generalized forecasting system. A significant portion of it relies on the particular strategies and tools developed by the Alberta Research Council's Atmospheric Sciences Department. However, we believe that our approach and methods provide some useful insights into developing systems with similar domain characteristics and design goals. The main issue for us has been organizing and representing a variety of different types of knowledge. Although METEOR does not have access to all the data available to human forecasters, its problem is still data intensive. METEOR has an internal representation of the forecast area that allows it to organize large quantities of meteorological measurements, synthesized information, and predictions. METEOR is also able to integrate this knowledge spatially and temporally. In this paper, we will focus on how we have approached this problem by integrating and extending two common AI formalisms: condition-action rules and frames.

II APPLICATION BACKGROUND

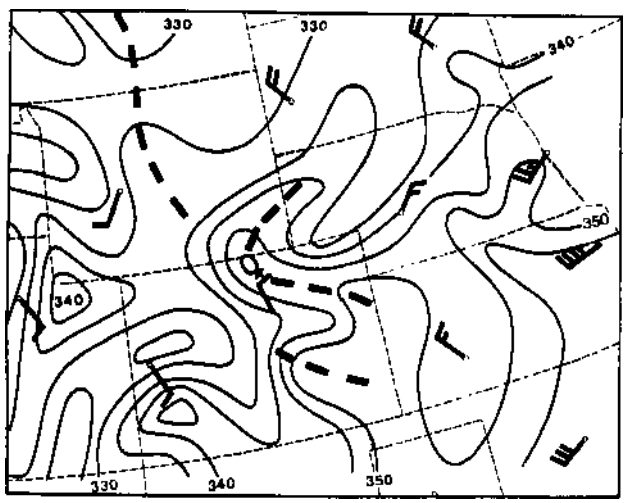
The Alberta Research Council's Atmospheric Sciences Department has conducted a research program on weather modification and hail suppression for a number of years. During "hail season," an experienced meterologist and several assistants are responsible for predicting the occurrence, severity, and path of hail storms for this research program.

The meteorologist begins his task by trying to understand patterns of meteorological activity. He typically consults a large number of maps, both diagnostic and prognostic in nature, generated from meteorological measurements taken at weather stations throughout the continent. These maps provide information such as temperature, humidity, and wind direction and speed at several levels of the atmosphere. He also uses a statistical index that has been developed to measure convective activity [1]. Very simply, thb index (called the Synoptic Index of Convection, or Sc4) combines four predictor variables aimed at

* This work was supported in part by an NSERC Industrial Research Fellowship to Renee Elio during a tenure at the Alberta Research Council.



(a) Sc4 Index Map



(b) Surface Moisture Index Map

Figure 1. Sample Index Contour Maps

evaluating whether the right ingredients—atmospheric instability and moisture—are present in the right amounts to generate convective activity. The index generates a single rating (-3 to +5) of the degree of potential convective activity. A higher rating means a more severe storm. For example, -1 means "scattered showers but no thundershowers" while +5 means "hail larger than golf balls". The positive-negative sides of this scale roughly correspond to a hail-no hail distinction.

This index is computed from meteorological data taken at the weather stations. Values are interpolated between stations and the final output is provided in the form of a contour map (Figure Ia). The expert interprets this map using other information. For example, he locates the regions of maximum Sc4 and considers wind speeds and directions at a particular pressure level. He then delineates a storm initiation region (the grey areas marked in Figure Ia) upwind of the maximum, where the values are changing most rapidly (i.e., areas of strong gradient

2006 YEG SA 0600 E50 BKN 90 OVC 15+TRW-
148/16/13/3010/996/ CB7AC3
(a)
LTGIC-CC-CG SW QUAD. SHWRS
HVIER N.W PRES UNSTDY 3012
(b)

Figure 2. Example station report for station YEG, with cloud cover report (a) and remarks section (b) marked.

on the map). The expert also produces another contour map of surface moisture (Figure Ib) and interprets it in a similar fashion. He combines the significant regions he has noted on each map to refine his prediction of storm location and to assess partially the direction of storm movement.

Other information from the weather station reports provide qualitative information on weather conditions that the expert can use to forecast. An example station report is given in Figure 2. A number of meteorological measurements are given in this report. The first, marked (a) in the figure, indicates which types of clouds are observed and what observable portion of the sky they cover. In this example, the first cloud layer the observer sees, cumulonimbus (CB), covers 7/10 of the sky. A second cloud layer of altocumulus (AC) covers the remaining 3/10 of the sky the observer can see. Other data in the report describe the state of this cloud cover. For example, the CB clouds are "broken" (BKN), meaning the observer can see through this layer. If an earlier report from this station had indicated the layer was overcast, this shift to broken might signify certain dynamics or processes were occurring.

The other important information is the optional remarks section of the report, marked (b) in the figure. In this section, the human observer at the weather station provides additional information on current conditions that are not easily expressed in coded format. What these remarks report is the following: "There is lightning in the clouds, from cloud to cloud, and from cloud to the ground in the southwest direction [relative to the station]. Showers are heavier in the north and west directions. The pressure [at the station] is unsteady."

Both the cloud cover information and the information contained in the remarks section have gone unanalyzed in the past because there is no simple way to "understand" their information and codify it in a form usable by statistical models. However, these qualitative conditions are meaningful to an experienced meteorologist familiar with this particular forecast area. For example, our expert has a number of informal heuristic rules based on cloud formations. An example from this "look-out-the-window" strategy is [a]If there are cirrostratus clouds, then it's not likely a storm will occur." This rule is not simply an empirical association, but is part of the expert's causal understanding of storm formation and thermodynamics.

In sum, this task is characterized by a huge amount of quantitative data that change during the course of the forecast period. The nature of how the data change over time is also informative. Meteorological measurements are usually displayed and interpreted graphically, and the forecaster often communicates

his own predictions and conclusions graphically as well. The forecaster's world and his knowledge have a four-dimensional quality: the "things" about which he reasons have spatial characteristics (in the horizontal and vertical dimensions) as well as temporal characteristics. Finally, the expert has qualitative knowledge and heuristics acquired from his experience with this forecast area. The use of this knowledge to interpret and augment the statistical indices distinguishes his forecasts from those of less experienced forecasters.

III KNOWLEDGE REPRESENTATION

As noted earlier, we will focus on our approach to representing the different types of knowledge in this data-intensive task. Since we believe no single formalism is appropriate for all types of knowledge that typically characterize expert problem-solving, our initial predisposition was to combine different knowledge representations. This turned out to be essential in handling this task. Several problem features were compatible with a rule-based approach: there were many sub tasks METEOR must perform to acquire and synthesize the data; there was a relatively well-defined strategic approach to interpreting the quantitative predictors and maps for making and fine-tuning predictions; and the qualitative knowledge could be captured in IF-THEN associations. On the other hand, much of the domain knowledge lends itself to a hierarchic organization that will permit inheritance of properties. For example, there is knowledge about the concept "cloud" that is shared by both "nonconvective clouds" and "convective clouds." There are many such hierarchically-related concepts, some related to weather conditions, others related to areas recognized on contour maps like those in Figure 1. More importantly, METEOR repeatedly instantiates these concepts dynamically for each forecast it makes during the day. The instantiated concepts must be organized in a way that is efficient for both storage and inferencing. We have chosen a frame-based representation for this type of knowledge.

Not all of METEOR's knowledge is symbolic, because not all of the data can be efficiently represented symbolically. The rule and frame components are augmented by LISP objects, primarily for representing and processing contour maps. The following sections concentrate on the manner in which the two primary formalisms were extended and integrated to represent the knowledge in this task.

A. Observations and Events

One unique demand of this task is its temporal quality. Meteorological events are processes occurring over time, but they are inferred from direct observations that are made at particular points in time. This is analogous to the distinction some expert medical diagnostic systems [2] make between symptoms and inferred pathological states. Understanding the relationship between observations and events is critical, because much of the expert's qualitative knowledge concerns the recognition of certain meteorological events and their implications for forecasting.

We distinguish between an observation and event in the knowledge representation, primarily in terms of their spatial and temporal characteristics. Observations occur at a particular place and time. Events, inferred from observations, range over time and space. While a set of observations may signal that a particular event is occurring, we cannot unequivocally know when the event really started or stopped from this single data point. Similar distinctions have been made by temporal logic systems [3]. For METEOR, the domain knowledge strongly constrains what is and is not likely temporally, so we use very simple temporal inference rules to capture the relation among observations and events.

Unlike observations, events can be separated into stages. Each stage specifies a set of required observations. More importantly, the stages indicate particular properties the required observations must have and how the observations must be temporally and spatially related. For example, the concept for the cloud type altocumulus-standing-lenticularis (ACSL) will be instantiated when ACLS is observed. An ACLS observation is one of the required observations for a LAMINAR-FLOW event. The event will be recognized, however, only when this ACLS observation occurs in the morning and to the west of certain other observations. In combination with a different set of observations, or with slightly different properties, this ACLS observation might signal another kind of event. The event frames specify these sorts of details about particular observation instantiations.

Knowledge relating observations and events is initially represented in frames, since this formalism provides an easily-understood specification of concepts and properties that are related through type hierarchies. However, recognizing that events are occurring from a set of observations is really a matching problem: the same *type* of observation might be shared by different events, and each event typically requires several different observations. Some production system languages, such as OPS [4,5], provide powerful matching algorithms which can be exploited for this kind of problem. Therefore, we separate event-matching into two stages: first find a set of potentially-matched events, and then check the observations against the detailed specifications on the event frames. The first stage is handled by production rules that METEOR automatically generates from specifications of required observations on event frames. In this way, METEOR takes advantage of the overlap among required observations associated with different events (via the rule matching network described in [5]) by building rules of the form *IF obs-1 obs-2 obs-4 ... THEN event-A* and *IF obs-1 obe-8 obs-4 ... THEN event-B.* When observations instantiated in the frame system are inserted in the production system's working memory, a number of potentially-matched events can be recognized. The second stage of event-matching examines each candidate event in turn, checking whether the relationships among instantiated observation match the specifications given in the candidate event's frame.

There are a number of advantages to this approach. First, it exploits the best aspects of the two formalisms. The frame structure allows instantiated concepts to participate in type hierarchies and have their interassociations available to METEOR as additional properties. Specific event properties can reference the properties of instantiated observations. Since searching for observations is a relatively time-consuming task, we do not want to keep looking for the same type of observation for different events. In this regard, recognizing when particular events are occurring is best done by production rule matching. By adhering to a particular format for specifying required observations on the event frame, we can modify or extend METEOR's knowledge about event and observation relationships

at the frame level and still ensure that legal production rules are generated, on an as-needed basis. Finally, the goal of event-matching is to be able to make additional inferences concerning storm development. The inference rules associated with events are also initially stored on the event frames. Only when an event is matched does METEOR migrate its inference rules from the frame representation to the rule representation.

The last important feature of events is their temporal quality. Because events are defined as spanning time, there must be a way to represent changes that a single property of an event can have over time. The most obvious example is "location". In general, any concept might have a mix of time-invariant and time-varying properties. We represent time-varying properties by specifying a path to a generic *time frame*. A time-frame essentially has an add/delete list format for indicating what a property's value was at a particular time. Time frames are used extensively throughout METEOR's knowledge representation. Not only do they provide a way of maintaining what properties were, but they serve as a consistent way of representing predictions as changes to properties for future times.

B. Intermingled Hierarchies

In this domain, the same concept can participate in several different conceptual hierarchies. However, some knowledge, especially membership, must be repeated across several hierarchies. We could use simple, unconnected hierarchies and duplicate knowledge when necessary, but there would be no easy way to maintain duplicated knowledge or to know all the hierarchies in which a concept participates. We have approached this problem by developing a *view* scheme that combines simple hierarchies into a single network. Repeated concepts in the original hierarchies become a single node in this new network. Each node represents the collection of all knowledge about a concept. Knowledge that was particular to a single hierarchy defines a certain view of the concept. This view is maintained as a further subdivision of knowledge within the node. Each node in the network has as many views as the number of simple hierarchies in which its concept originally participated. Not all knowledge about a concept has to be associated with a view—it is possible to declare knowledge as being true of the concept in general. This approach is similar in spirit, if not complexity, to previous proposals for organizing knowledge around different perspectives and viewpoints [6].

An example is illustrated in Figure 3. In this case, the concept CLOUD is a member of both the OBS (observation) hierarchy and EVENT hierarchy. The complete concept for CLOUD includes both its views as an OBS and an EVENT. Knowledge generally true of CLOUD (that would have been found in both the OBS and EVENT hierarchies) resides on the CLOUD frame (frame 3). Knowledge unique to a CLOUD view (that would have been found in only one of the original hierarchies) is represented in view frames (frames 4 and 5) associated with the main node frame (frame 3). There are three ways to separate the total knowledge about CLOUD. It can be viewed as an EVENT, in which case knowledge is collected from frames 4, 3, and 1. When viewed as an OBS, knowledge from frames 5,3, and 2 is accessible. Finally, all the knowledge associated with CLOUD (frames 1-5)—previously split over different hierarchies—is also accessible from CLOUD. This includes knowledge about what views it has. It might seem that all this knowledge could reside on the single CLOUD frame, with the property names implying

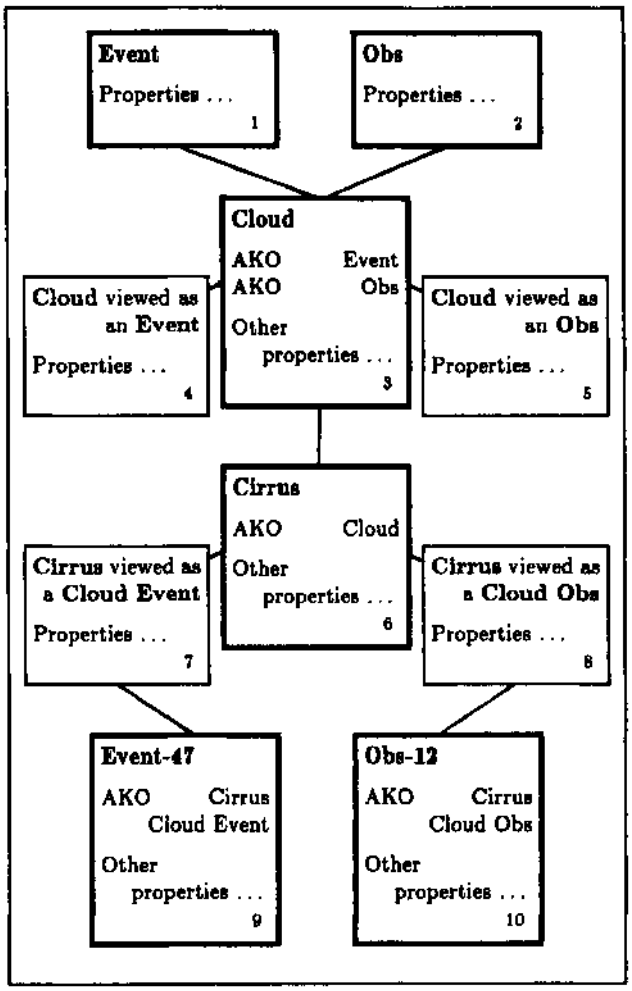


Figure 3. Views in Two Combined Hierachies

the hierarchy from which each piece of knowledge originated. However, it is possible that the same property of a concept could have different values in different hierarchies. By placing properties on distinct views, we avoid this potential clash.

It is not necessary for a concept to have more than one parent in order to have views. Although CIRRUS has only one parent in this example network, it was repeated in the original unconnected hierarchies. So, just like CLOUD, CIRRUS has views that reflect the original hierarchies. For example, CIRRUS can be viewed as a CLOUD EVENT, in which case knowledge is collected from frames 7, 6, 4, 3, and 1. When viewed as a CLOUD OBS, knowledge is collected from frames 8, 6, 5. 3, and 2. In effect, CIRRUS inherits views (e.g., frames 4 and 5) from its parent, and consequently inherits properties from those views.

Frames 1-8 in Figure 3 represent part of METEOR's initial hierarchy. The concepts at the lowest level of this hierarchy can be dynamically instantiated. It is possible to instantiate a particular view of a concept, thereby restricting the inheritance of other views associated with that concept. Frames 9 and 10 illustrate two examples of view instantiations—EVENT-47 and

OBS-12. (These event and observation instantiations do not occur simultaneously and do not necessarily co-exist.) OBS-12 is an instantiation of CIRRUS viewed as a CLOUD OBS. It inherits only those properties on an inheritance path that include frames 8, 6, 5, 3, and 2. Similarly, EVENT-47, as a-kind-of CIRRUS CLOUD EVENT, inherits only those properties on a path consisting of frames 7, 6, 4, 3, and 1.

The main advantages to this scheme are (a) properties common to all views can be shared; (b) the structure of the network need not be repeated for every simple hierarchy; (c) the number of ways of viewing a particular concept is part of the knowledge available to the system; and (d) the property inheritance mechanism of simple hierarchies is preserved.

IV IMPLEMENTATION DETAILS

Figure 4 shows the current METEOR system and its major parts. METEOR runs on a Xerox 1100 Lisp Machine and is written in INTERLISP-D and OPS4 [4]. Initial data collection is done on a VAX 11/780 running VMS.

A. VAX 11/780 Side

METEOR is configured by information that resides on the VAX. This information contains geographical information about the forecast area, weather stations in the area, and its division into larger geographical regions (described below). It also describes a directory that is used to organize both station reports and contour maps generated from this station data. Data from weather stations (Figure 2) arrives hourly on a dedicated circuit line. This station data is automatically collected by processes that monitor a particular set of stations in the forecast area (currently about 70 stations). At present, METEOR is being tested and refined using several test cases from the 1984 hail season.

B. Xerox 1100 Side

The frame and rule-based components reside on the 1100. The rule-based component is currently implemented as three OPS production systems (although they are compiled together). Our version of OPS4 was extended to compile its action sides, and its interface with the INTERLISP-D environment was improved. The Forecaster production system initiates the configuration of the METEOR'S "storm world" (described below) and is responsible for data acquisition, interpretation, and forecasting subtasks. This Forecaster production system invokes two other production systems that have very specialized tasks: the Remarks Parser production system, which "translates* and parses the human observations contained in the station report remarks (Figure 2) and the Region Builder production system, which identifies regions of meteorological activity from the parsed remarks and cloud reports (Figure 2). These are described in more detail below.

The frame and view component was implemented in INTERLISP-D. The production systems interact with the frame representation via the flexibility afforded by the OPS4 language in accessing the INTERLISP-D environment. Thus, the production rules can examine frames, transfer knowledge from frames into working memory, build new rules from knowledge stored on frames, initiate processes for building maps on the VAX and transferring them to the 1100, and initiate the creation of new

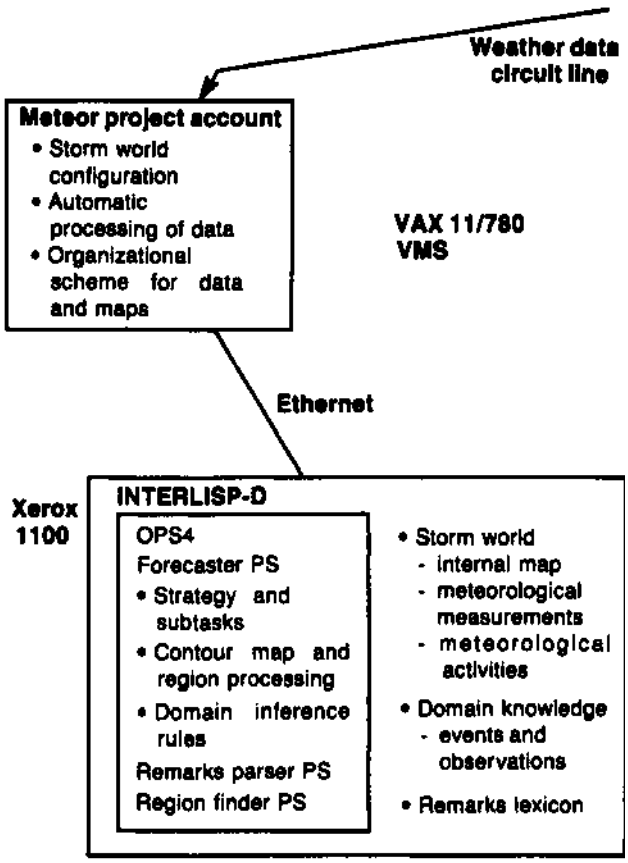


Figure 4. System Organization

frames to represent the instantiation of particular meteorological activities it has derived or inferred.

METEOR currently starts with 140 production rules (across the three PSs) and 300 frames. In the course of forecasting, many new frames are dynamically created that represent particular instantiations of the concepts in the type hierarchies. This could be an additional 300 or more frames per forecast. (There are typically four forecasts per day.) New OPS rules are dynamically generated from information stored in frames on an as-needed basis, particularly to aid the event-matching described earlier.

C. METEOR Output

METEOR outputs it predictions by displaying a map of the forecast area on the 1100 screen and outlining the location of one or more storm initiation regions. For each region, METEOR displays a predicted intensity and direction, as well as the factors that support these predictions. Qualitative influences on these predictions are also reported, e.g., "intensity may be lessened due to the influence of the mountains."

V METEOR'S STORM WORLD

Given maps like those in Figure 1, METEOR must be able to locate areas of absolute or relative maximum, identify areas of strong contour gradient, combine them with the location of other spatially-defined regions, and create new regions. There

are a number of questions implicitly asked and answered in this process: Does this region intersect with another region of a particular type? Was there a particular activity southeast of this region two hours ago? Thus, a knowledge-representation framework must support creating and reasoning about entities defined in space and time.

A. Representing Meteorological Activities

METEOR's storm world (about 2.3 million km^2) is represented as a two-dimensional array of map elements or *mapels*. Mapels have no meteorological significance. The location of all objects in this world are defined by mapels. There are three major types of objects in this world: STATIONS, REGIONs, and GEOREGIONs. A STATION is a source of data, located by a single mapel. A REGION is a collection of mapels to denote some particular meteorological activity. Example types of regions are "Sc4 maximum", "storm initiation", or "cirrostratus observation". GEOREGIONs are collections of mapels, larger than REGIONs, that represent a higher-level organization of the storm world. They are used primarily to speed certain inferences about the relative locations of regions. An example georegion would be "Central Alberta". The knowledge associated with these objects and their members is represented in frames within a type hierarchy. While STATION and GEOREGION have a fixed set of members, REGION is dynamically instantiated to represent areas on contour maps and meteorological activities that are located in space and time. In fact, all METEOR's dynamic instantiations must be a-kind-of REGION and have a corresponding a view as a REGION. In Figure 3, then, the OBS-12 instantiation of *CIRRUS viewed as a CLOUD OBS* would more commonly be represented as concept with two separate views: the view shown in Figure 3 as frame 10 and another view as a REGION.

The properties that REGION instantiations inherit from REGION are primarily spatial in nature. One such property is the *mapels* property. While this property is time-invariant for STATION and GEOREGION members (since these objects do not move), it is time-varying for some instantiations of REGION (for activities moving through the forecast area). The time frames that describe the dynamic properties that OBS-12 inherits as a-kind-of REGION are linked to its view as a REGION. By using time frames to describe the dynamic properties of the entire storm world (e.g., what kinds of activities populated the world at a particular time), METEOR can keep track of what the world looked like in the past and what it predicts the world might look like in the future. Under this scheme, answers to the kinds of questions given above ("Does Region A intersect with Region B?" and "Was there a Sc4 maximum southwest of Region C four hours ago?") are handled essentially the same way by the same mechanisms.

B. Building Regions

The production systems direct the identification and creation of new regions from two types of raw data: contour maps (Figure 1) and weather station reports (Figure 2).

1. Regions from Contour Maps—To glean the same information from the Figure 1 contour maps as the expert does, METEOR must recognize areas of maxima, locate the interesting areas of strong contour gradient, and create new entities in its internal storm world corresponding to these areas. Plotted contour maps like those in Figure 1 can be produced on the VAX and transferred to the 1100 at METEOR's request.

The expert's strategy for locating "interesting" gradients on the contour map is embodied in the Forecaster production system. The potentially interesting gradients are those with the greatest changes in slope that occur upwind of the maximum. The Forecaster production system consults wind speed and direction data stored in the frame system (a time-varying property of certain STATION members). To get the changes in the slope of the contours, METEOR takes the first derivitive of the contour map along the direction of the proper wind. Given the original contour map, a derivitive map, and the mapel-space map, METEOR examines those portions of the derivitive map that are upwind of the line of maximum. Candidate storm initiation regions are strong gradients (those with a high enough derivitive value). Once regions are located in mapel space, METEOR can create regions in its internal storm world like the grey areas in Figure 1 outlined by the forecaster.

2. Regions from human remarks—There is considerable variability in the nature of abbreviations, in whether punctuation is used to join elements or signal the end of a phrase, and in word order. This variability occurs both within and across remarks. The job of METEOR's Remarks Parser production system module (a production system plus lexical knowledge represented in frames) is to extract the remark from the station data, translate the abbreviations into full words, and parse the remark into its proper constituents. The Remarks Parser relies solely on a set of syntactic heuristics gleaned from a corpus of remarks.

The parsed remarks are a time-varying property of the station from which they originated. The Qualitative Region PS can then satisfy a request from the Forecaster PS such as "determine if there are cirrostratus observations in northwestern Alberta." This request typically occurs when the Forecaster production system is involved with event-matching. The request will actually include detailed property specifications for an observation that were found on the event's frame. The Qualitative Region PS examines the parsed remarks as well as the cloud cover information, searching for the requested observations, and locates those finds in mapel space. It then computes a set of mapels to define that region, and creates a concept that has views as a REGION and as particular kind of OBServation.

VI RULE JUSTIFICATION

In this paper, we have concentrated on describing how METEOR's knowledge was designed to accommodate the problem-solving demands of this task. We want to describe briefly a further use of this framework for investigating how procedural and declarative representations can be integrated to obtain more expert-like problem solving. One feature of expert problem solving is the "conceptual leaps" that an expert seems to make. This observation that much of an expert's knowledge seems "compiled" [7,8] is a notion common to many theories of skilled performance. There is usually a chain of causal relationships and supporting inference rules connecting what otherwise appears to be a mysterious transition from an antecedant condition to a consequence. We recognize this when our expert, having given a rule such as "If there are cirrostratus clouds, then a severe storm is unlikely," can produce a detailed causal justification for this assertion.

As noted earlier, METEOR has a redundant representation of some domain knowledge: those relating events and observation types, and those asserting the implications of a matched event. This knowledge resides in the event's frame representation but is migrated to the rule representation on an as-needed basis. We are exploring this method as a means of unpacking inference rules by referencing causal connections and relationships among the rule's related frames. Just as an event's *required-observations* property specifies particular values for properties of instantiated observations, an event's inference rules are stored as properties whose values reference other concepts in the network. A mechanism aimed at justifying a procedurally-represented rule can follow these references through the frame representation. In doing so, it will find a number of low-level inference rules and relations that were never represented procedurally or directly used in problem-solving. These low-level rules and relations are the causal underpinnings of the expert's conceptual leap.

For example, the justification for the cirrostratus rule above rests on the notion that the required "players" associated with a cirrus event (e.g., vertical motion in a downward direction) have properties that clash with the required "players" for a storm event (e.g., vertical motion in an upward direction). These players can also exist as concepts in the type hierarchies, with their own associated inference rules indicating how instantiations of their properties would effect the properties of other concepts. For instance, If VERTICAL-MOTION's *direction* property were instantiated as *down,* this would suggest that RELATIVE-HUMIDITY's *direction-of-change* property should be instantiated as *decreasing).* The notion of required players assembling with the correct properties and affecting each other's the properties is similar to some ICAI approaches aimed at helping students understand complex causal processes [9].

By separating the procedural knowledge from its declarative support, one can have an expert system that problem-solves as if its knowledge were compiled, but that could back up that compiled knowledge on demand. By having a facility for automatically migrating declaratively-represented rules to procedural form during problem solving, an expert system could access a wider range of inference rules on an as-needed basis than the set it originally began the problem with.

VII SUMMARY AND CONCLUSIONS

METEOR could have been designed to take symbolic specifications of weather conditions, along with their spatial and temporal characteristics, from a user. However, this would have precluded using much of the data we currently interpret. The quantity of the data, particularly qualitative information on weather conditions, was so great that it could not be processed by human forecasters in real-time. In this sense, METEOR goes beyond what the expert currently does. This has increased the complexity of the problem, since this is the first opportunity the expert has had to use his qualitative knowledge about ongoing weather conditions during real-time forecasting. The positive aspect is that METEOR will not only be useful to less-experienced forecasters, but will provide the expert with additional information that has been intelligently incorporated into its forecast.

We found it necessary to use and extend a number of different formalisms to handle the large amount of different types of knowledge: the meteorological measurements, data in map form, hierarchically-related domain concepts, strategies and subtasks, and qualitative inference rules. The view scheme is an efficent method of organizing the knowledge METEOR uses. It also imposes a rigor on the frame structure and has made knowledge about this structure more explicit to the system. By giving some procedural knowledge an initial declarative representation, we have made this knowledge more accessible and potentially more versatile. These redundant representations could facilitate rule justification without compromising the conceptual-leap aspect of expert problem-solving. Why, when, and how an expert system might unpack its procedural knowledge to aid its own problem-solving requires further consideration of the role of causal models in expert reasoning.

VIII ACKNOWLEDGMENTS

We are grateful to Geoff Strong, our expert, for his continued patience and cooperation throughout this project.

IX REFERENCES

Strong, G. S. & Wilson, W. D. The synoptic index of convection: Application to the Fort Collins Hailstorm of 30 July 1979. Paper presented at the American Meterological Society Conference, Tulsa, 1983.

[2] Weiss, S.M., Kulikowski, C.A., Amarel, S., and Safir, A. A model-based method for computer-aided medical decision-making. *Artificial Intelligence,* 1978, *11,* 145-172.

Vilain, M. B. A system for reasoning about time. *Proceedings of the National Conference on Artificial Intelligence,* 1982, 197-201.

Forgy, C. L. The OPS4 Reference Manual. Technical Report, Department of Computer Science, Carnegie-Mellon University, 1979.

Forgy, C. L. Rete: A fast algorithm for the many pattern/ many object pattern match problem. *Artificial Intelligence,* 1982, *19,* pp 17-37.

Bobrow, D. G. & Winograd, T. An overview of KRL, a knowledge representation language. *Cognitive Science*, 1977, 1, pp 3-46.

[7] Szolovits, P. & Pauker, S. G. Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence,* 1978, *11,* 115-144.

[8] Clancy, W.J. The epistemology of a rule-based expert system: A framework for explanation. Stanford University Technical Report No. STAN-CS-81-896, 1981.

[9] Stevens, A. L., Collins, A., &. Goldin, S. Diagnosing students' misconceptions in causal models. BBN Report # 3786, 1978, Bolt, Beranek and Neuman, Inc., Cambridge, Mass.