RUM: A Layered Architecture for Reasoning with Uncertainty

Piero P. Bonissone and **Steven S. Gans** and **Keith S. Decker**
GE Corporate Research and Development Center
P.O. Box 8
Schenectady, New York 12301*

Abstract

New reasoning techniques for dealing with uncertainty in expert systems have been embedded in RUM, a Reasoning with Uncertainty Module. RUM is an integrated software tool based on a frame system (KEE) that is implemented in an object oriented language. RUM's capabilities are subdivided into three layers: *representation, inference*, and *control.*

The representation layer is based on frame-like data structures that capture the uncertainty information used in the inference layer and the uncertainty meta-information used in the control layer. Linguistic probabilities are used to describe the lower and upper bounds of the certainty measure attached to a well formed formula. The source and the conditions under which the information was obtained represent the non-numerical meta-information.

The inference layer provides the uncertainty calculi with which to perform the intersection, detachment, union, and pooling of information. Five uncertainty calculi, based on their underlying Triangular norms, are used in this layer.

The control layer uses the meta-information to select the appropriate calculus for each context and to resolve eventual ignorance or conflict in the information. This layer enables the programmer to declaratively express the local (context dependent) meta-knowledge that will substitute for the global assumptions traditionally used in uncertain reasoning.

RUM has been tested in a sequence of experiments in both naval and aerial situation assessment, consisting of correlating reports and tracks, locating and classifying platforms, and identifying intents and threats.

I. Introduction

In most realistic situations, the information available to the decision maker is incomplete and uncertain. In automated reasoning systems, these two facets of the information have usually been treated independently. Theories and techniques for dealing with incomplete (but precise) information have evolved into the development of non-monotonic logics, truth maintenance systems (TMS) and reason maintenance systems (RMS). Theories and techniques for dealing with uncertain (but complete) information have either been adapted from other fields such as probability theory, based on unrealistic global assumptions, or proposed as *ad hoc* solution without formal justifications [Bonissone, 1987].

The trend followed by most approaches for reasoning with uncertainty has shown an almost complete disregard for the fundamental issues of automated reasoning, such as the proper *representation* of the information, the allowable *inference* paradigms suitable for the representation, and the *control* of such inferences. The majority of the approaches to reasoning with uncertainty do not properly cover these issues. Some approaches lack expressiveness in their representation paradigm. Other approaches require unrealistic assumptions to provide uniform combining rules defining the plausible inferences.

Specifically, the non-numerical approaches [Cohen and Grinberg, 1983b, Cohen and Grinberg, 1983a, Doyle, 1983] are inadequate to represent and summarize measures of uncertainty. The numerical approaches, on the other hand, generally tend to impose some restrictions upon the type and structure of the information (e.g., mutual exclusiveness of hypotheses, conditional independence of evidence). Most numerical approaches represent uncertainty as a precise quantity (scalar or interval) on a given scale. They require the user or expert to provide a *precise* yet *consistent* numerical assessment of the uncertainty of the atomic data and of their relations. The output produced by these systems is the result of laborious computations, guided by well-defined calculi, and appears to be equally precise. However, given the difficulty in consistently eliciting such numerical values from the user, it is clear that these models of uncertainty require an unrealistic level of precision that does not actually represent a real assessment of the uncertainty.

With few exceptions, such as MRS [Genesereth, 1982], the control of the inference process in most expert systems has been procedurally embedded in the inference engine, thus preventing any opportunistic and dynamic change in ordering inferences and in aggregating uncertainty. Usually, the same set of aggregation operators (i.e., the same uncertainty calculus) is selected *a priori* and is used uniformly for any inference made by the expert system. In the few numerical approaches where conflicting information is detected [Shafer, 1976], conflict handling is done in the inference layer, where the conflict resolution procedure is embedded in the same com-

*This work was partially supported by the Defense Advanced Research Projects Agency (DARPA) under USAF/Rome Air Development Center contract F30602-85-C-0033. Views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official opinion or policy of DARPA or the U.S. Government.

bining rules. This procedure consists of simply removing the conflicting part of the information. The non-conflicting portion is then normalized and propagated as if the conflict never existed.

It is our claim that the formalism for reasoning with uncertainty must exhibit the same structural (layered) decomposition typical of other automated reasoning methodologies, i.e., it must address each of the three layers of *representation, inference,* and *control.* The formalism must also be based on sound theoretical foundations to guarantee its general applicability to a variety of reasoning tasks. The proposed layered approach will be suitable to integration with reason maintenance systems that provide a distinction between the object logic theory (inference layer) and the meta logic theory (control layer).

This paper describes the theory, design, implementation, and testing of RUM, a Reasoning with Uncertainty Module, whose layered architecture reflects the above concerns. The next two sections (2 and 3) summarize RUM's underlying theory and design. The remaining two sections (4 and 5) describe an experiment in situation assessment used to test RUM and present the conclusions of this work.

II. RUM's Underlying Theory

Preliminary theoretical results were presented in two previous publications [Bonissone and Decker, 1986, Bonissone, 1986]. This section summarizes some of those results and provides a unified framework for their interpretation and use in RUM's architecture. A philosophical motivation for the RUM's three layer organization can also be found in [Bonissone, 1987].

A. Term Sets of Linguistic Probabilities

In expert system applications, users and experts must frequently provide subjective assessments of probability. Due to the difficulty of eliciting precise and consistent numerical certainty values, we have suggested the use of term sets of linguistic probability. Each term set determines the *granularity* (the finest level of specificity) of the measure of certainty that the user/expert can *consistently* provide.

A term set of linguistic probabilities is the set of symbols $\mathbf{L} = L_1, L_2, \ldots, L_n$. The meaning of each term $L_i \in \mathbf{L}$ is represented by a fuzzy number on the [0,1] interval. A computationally efficient way to characterize a fuzzy number is to use a parametric representation of its membership function. This parametric representation [Bonissone, 1982, Bonissone and Decker, 1986] is achieved by the 4-tuple $(a_i, b_i, \alpha_i, \beta_i)$. The first two parameters indicate the interval in which the membership value is 1.0; the third and fourth parameters indicate the left and right *width* of the distribution. Linear functions are used to define the slopes. Therefore, the membership function $\mu_{L_i}(x)$ is defined as

$$\mu_{L_i}(x) = \begin{cases} 0 & \text{if } x < (a_i - \alpha_i) \\ (\alpha_i^{-1})(x - a_i + \alpha_i) & \text{if } x \in [(a_i - \alpha_i), a_i] \\ 1 & \text{if } x \in [a - i, b_i] \\ (\beta_i^{-1})(b_i + \beta_i - x) & \text{if } x \in [b_i, (b_i + \beta_i)] \\ 0 & \text{if } x > (b_i + \beta_i) \end{cases}$$

For compactness of notation, we will denote the meaning of the term set element L_i parametrically. Table 1 illustrates one of four default RUM term sets, the nine element **L-nine**.*

Index	Symbol	Meaning
1	*impossible*	(0 0 0 0)
2	*extremely-unlikely*	(.01 .02 .01 .05)
3	*very-low-chance*	(.1 .18 .06 .05)
4	*small-chance*	(.22 .36 .05 .06)
5	*it-may*	(.41 .58 .09 .07)
6	*meaningful-chance*	(.63 .80 .05 .06)
7	*most-likely*	(.78 .92 .06 .05)
8	*extremely-likely*	(.98 .99 .05 .01)
9	*certain*	(1 1 0 0)

Table 1: The Nine Element Term Set L-nine

RUM's representation layer allows the user to characterize the lower and upper bounds of the certainty of a fact, or the sufficiency and necessity of a rule, by using elements of a selected term set.

B. Triangular Norms

Triangular norms (T-norms) and Triangular conorms (T-conorms) are the most general families of binary functions that satisfy the requirements of the conjunction and disjunction operators, respectively [Bonissone and Decker, 1986]. A T-norm is defined as a mapping $T : [0,1]^2 \rightarrow [0,1]$ which is monotonic, commutative, and associative. The boundary conditions of a T-norm satisfy the truth tables of the logical AND operator. The T-conorms are defined in terms of the T-norms and a negation operator, by using a generalization of DeMorgan's duality. Thus, for a suitable negation operator, such as $N(a) = 1 - a$, the T-conorm $S(a, b)$ is defined as

$$S(a, b) = N(T(N(a), N(b)))$$

We have seen that the use of term sets determines the granularity with which the input certainty is described. This granularity limits the ability to differentiate between two similar calculi; only a finite, small subset of the infinite number of calculi that can be generated from a parametrized T-norm family produces notably different results.

This result has been confirmed by an experiment [Bonissone and Decker, 1986] where eleven different calculi of uncertainty, represented by their corresponding T-norms, were analyzed. The experiment showed that five equivalence classes were needed to represent (or reasonably approximate) any T-norm. The corresponding five uncertainty calculi were defined by the common negation operator $N(a) = 1 - a$ and the DeMorgan pair $(T_{Sc}(a, b, p), S_{Sc}(a, b, p))$ for the following values

*The meanings associated with each element of the term set were derived from the results of psychological experiments on the use of linguistic probabilities [Beyth-Marom, 1982].

$p = -1$	$T_1(a,b) = \max(0, a+b-1)$	$S_1(a,b) = \min(1, a+b)$
$p = -0.5$	$T_{Sc}(a,b,-0.5) = \max(0, a^{0.5}+b^{0.5}-1)^2$	$S_{Sc}(a,b,0.5) = 1-\max(0,((1-a)^{0.5}+(1-b)^{0.5}-1))^2$
$p \to 0$	$T_2(a,b) = ab$	$S_2(a,b) = a+b-ab$
$p = 1$	$T_{Sc}(a,b,1) = \max(0, a^{-1}+b^{-1}-1)^{-1}$	$S_{Sc}(a,b,1) = 1-\max(0,((1-a)^{-1}+(1-b)^{-1}-1))^{-1}$
$p \to \infty$	$T_3(a,b) = \min(a,b)$	$S_3(a,b) = \max(a,b)$

Table 2: Five Uncertainty Calculi

of p in Table 2:**

RUM's inference layer provides the user with a selection of the five T-norm based calculi described above.

III. RUM's Layered Architecture

RUM's architecture is based on three layers: *representation, inference*, and *control*. The first layer (the representation layer) includes the structure required to capture information used in the inference layer and meta-information used in the control layer. In this structure, linguistic probabilities are used to describe the lower and upper bounds of the certainty measure associated with a well-formed formula (*wff*). Non-numerical meta-information, describing the source and the conditions under which information was obtained, is represented in this layer along with numerical meta-information describing the amount of ignorance and consistency.

The second layer (the inference layer) includes five uncertainty calculi based on their underlying Triangular norms (T-norms). Any operation required by an uncertainty calculus can be expressed in terms of its T-norm and a negation operator. Note that T-norm-based calculi have various computational advantages: they are *truth-functional, commutative*, and *associative*. Therefore, if numerical computations to evaluate T-norm-based expressions are carried out at run-time, the above properties ensure that any result can be directly computed from the individual value of each argument; that the result is independent from the order of the arguments; and that for more than two arguments, the evaluation of T-norm expressions can be done recursively.

The third layer (the control layer) includes the functions required to select the calculus appropriate for each context and to resolve ignorance or conflict in the information. These functions rely on *local* (i.e., context-dependent) knowledge about the information (*meta-knowledge*). The scope of the calculus selection and ignorance/conflict resolution is limited to the context (knowledge base subset) for which the meta-knowledge is available. Figure 1 illustrates RUM's architecture. The following sections describe RUM's functions attached to each of the three layers.

**$T_{Sc}(a,b,p)$ is one of six parametrized families of T-norms discussed in [Bonissone and Decker, 1986]. It was originally defined by Schweizer and Sklar [Schweizer and Sklar, 1963] and exhibits both complete coverage of the T-norm space and numerical stability.

A. The Representation Layer

1. RUM's Wff System

RUM is an integrated software tool based on a frame system (KEE™) [1986] that is implemented in an object-oriented language. RUM's Wff System modifies KEE's representation of a *wff*. RUM's *wff* is the pair (<unit> <slot>), which is the description of a variable in the problem domain. For each *wff*, a corresponding certainty unit is created. The unit contains a list of the considered values that the variable may take, and for each of these values it maintains the *lower* and *upper bounds* of its certainty, an *ignorance measure*, a *consistency measure*, and the *evidence source*. The ignorance measure is computed as the area of the fuzzy interval formed by the lower and upper bounds, while the consistency measure checks to see if the lower and upper bounds have crossed.

RUM's Wff System allows the user to express arbitrary uncertainty granularity by providing the flexibility to mix precise and imprecise measures of certainty (crisp numbers or intervals, fuzzy numbers or intervals, linguistic values) in defining both the input certainties and the rule strengths.

2. RUM's Rule System

RUM's Rule System replaces KEE Rule System-3 capabilities by incorporating uncertainty information in the inference scheme. The uncertain information is described in the certainty units of the *wffs*, represented in RUM's Wff System, and in the degrees of necessity and sufficiency attached to each rule. A rule is represented by a frame with several slots. These slots include the name of the rule; the lists of contexts, premises, and conclusions; the rule's sufficiency and necessity; and the T-norm to be used for aggregation. All slots (except the name, premises, and consequences) have default values. The contexts, premises, and conclusions can comprise values, variables, RUM predicates and arbitrary LISP functions. Rules with unbound variables are instantiated with the necessary environment to produce rule instances.

The T-norm specified with the rule is used to aggregate the certainties of the rule premises and to perform detachment (which computes the certainty of the conclusion given the sufficiency and necessity of the rule). It defaults to T_3, which is the MIN function. The associated T-conorm is used to aggregate the certainties of identical conclusions inferred by multiple rule instances derived from the *same* rule. These are often subsumptive, and the value defaults to S_3, the MAX function. Finally, each seperate consequence of a rule has a specified T-conorm that will be used to aggregate the consequence with

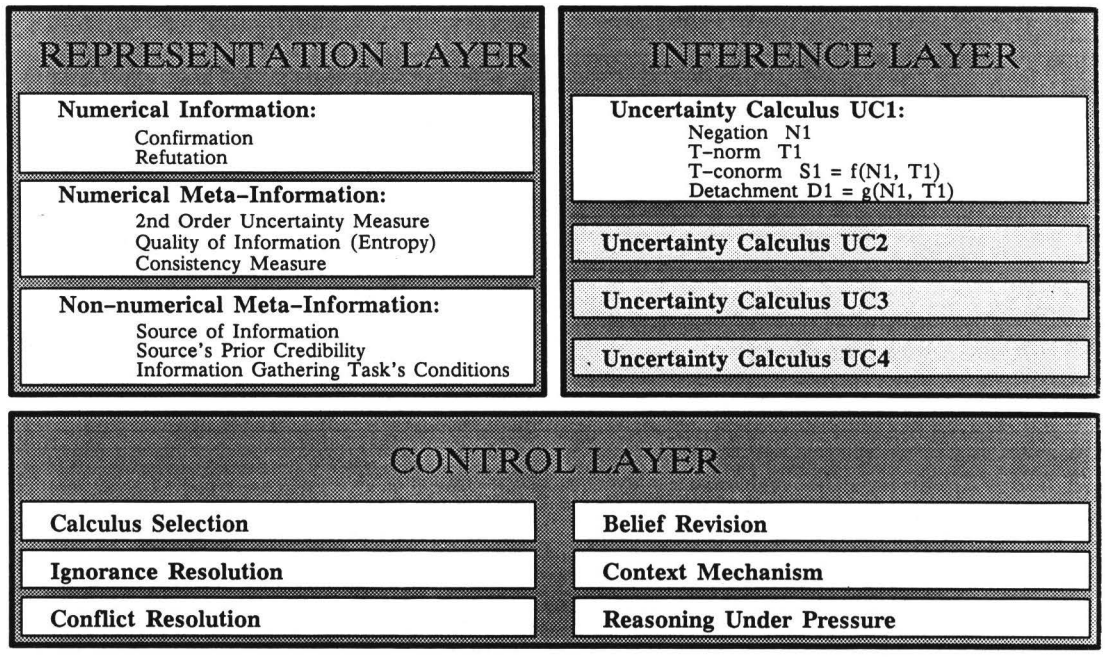

Figure 1: RUM's Three Layer Architecture

identical consequences derived from *different* rules.

B. The Inference Layer

1. Calculi Operations

For each calculus, four operations are defined in RUM's Rule System: *premise evaluation, conclusion detachment, conclusion aggregation,* and *source consensus*. Each operation in a calculus can be completely defined by a Triangular norm and a negation operator (in logic any boolean expression can be rewritten in terms of an intersection and complementation operator). The four operations are defined as follows:

Premise evaluation: The premise evaluation operation determines the degree to which all the clauses in the rule premise have been satisfied by the matching *wffs*. Let b_i and B_i indicate the lower and upper bounds of the certainty of condition i in the premise of a given rule. Then the aggregated premise certainty range $[b, B]$ is defined as

$$[b, B] = [T(b_1, b_2, \ldots, b_m), T(B_1, B_2, \ldots, B_m)]$$

Conclusion Detachment: The conclusion detachment operation indicates the certainty with which the conclusion can be asserted, given the strength and appropriateness of the rule. Let s and n be the lower bounds of the degree of *sufficiency* and *necessity*, respectively, of the given rule, and let $[b, B]$ be the computed premise certainty range. Then the range $[c, C]$, indicating the lower and upper bound for the certainty of the conclusion inferred by such a rule, is defined as

$$[c, C] = [T(s, b), N(T(n, N(B)))]$$

The degrees of sufficiency and necessity, respectively, indicate the amount of certainty with which the rule premise implies its conclusion and vice versa. The sufficiency degree is used with *modus ponens* to provide the lower bound of the conclusion. The necessity degree is used with *modus tollens* to obtain a lower bound for the complement of the conclusion (which can be transformed into an upper bound for the conclusion itself).

Conclusion aggregation: The conclusion aggregation operation determines the consolidated degree to which the conclusion is believed if supported by more than one path in the rule deduction graph, i.e., by more than one rule instance. It is also possible to have various groups of deductive paths, i.e., various sets of rule instances, all supporting the same conclusion. Each group of deductive paths can have a distinct conclusion aggregation operator associated with it. Let the ranges $[c_j, C_j]$ indicate the certainty lower and upper bounds of the *same* conclusion inferred by various rule instances belonging to the same group. Then, for each group of deductive paths, the range $[d, D]$ of the aggregated conclusion is defined as

$$\begin{aligned}[d, D] \;=\; & [N(T(N(c_1), N(c_2), \ldots, N(c_m))), \\ & T(N(C_1), N(C_2), \ldots, N(C_m))]\end{aligned}$$

RUM distinguishes between rule instances generated from the same rule and rule instances derived from different rules. Rule instances generated from the same rule are aggregated first, to take into account the normally large amount of redundancy that such instances entail. Rule instances derived

from different rules are subsequently aggregated taking into account the knowledge about the presence or lack of positive or negative correlation that characterizes the various rules.

Source Consensus: The source consensus operation reflects the fusion of the certainty measures of the same evidence A provided by different sources. The evidence can be an *observed* fact, or a *deduced* fact. In the former case, the fusion occurs before the evidence is used as an input in the deduction process. In the latter case, the fusion occurs after the evidence has been aggregated by each group of deductive paths. The source consensus operation reduces the ignorance about the certainty of A, by producing an interval that is always smaller or equal to the smallest interval provided by any of the information sources. If there is an inconsistency among some of the sources, the resulting certainty intervals will be disjoint, thus introducing a conflict into the aggregated result. Let $[L_1(A), U_1(A)], [L_2(A), U_2(A)], \ldots, [L_n(A), U_n(A)]$ the certainty lower and upper bounds of the same conclusion provided by different sources of information. The result $[L_{tot}(A), U_{tot}(A)]$, obtained from *fusing* all the assertions about A, is calculated by taking the intersection of the certainty intervals:

$$[L_{tot}(A), U_{tot}(A)] = [\max_i(L_i(A)), \min_i(U_i(A))]$$

C. The Control Layer

1. Calculi Selection

RUM's Rule System uses a set of T-norm-based calculi to handle uncertain information. The calculus used by each rule instance is inherited from the rule subclass. The calculus can be modified through KEE's user interface or programmatically (e.g., by an active value or a task, see Section 4.). Class inheritance can also be used to modify the degree of sufficiency and necessity of all the rule members of the same class.

The calculi selection process consists of two assignments. The first assignment indicates the T-norm with which the premise evaluation and the conclusion detachment will be computed. Such an assignment is made for each rule and is passed (through inheritance) to all rule instances derived from a rule.

The second assignment indicates the T-conorm (represented by its dual T-norm) with which the conclusion aggregation will be computed. This assignment is made for each subset of rule instances generated from *different* rules that assert the same conclusion.

The characteristics of a T-norm will determine how it is used. The T-norm assigned to each rule for premise evaluation and conclusion detachment will be a function of the decision maker's *attitude toward risk.* The ordering of the T-norms, which is identical to the ordering of parameter p in the Schweizer and Sklar family of T-norms, reflects the ordering from a conservative attitude ($p = -1$ or T_1) to a non-conservative one ($p \rightarrow \infty$ or T_3). From the definition of the calculi operations, we can see that T_1 will generate the smallest premise evaluation and the weakest conclusion detachment (i.e., the widest uncertainty interval attached to the rule's conclusion.) T-norms generated by larger values of p will exhibit less drastic behaviors and will produce nested intervals with their detachment operations. T_3 will generate the largest premise evaluation and the strongest conclusion detachment (the smallest certainty interval).

For the second assignment, the T-norm that aggregates the subsets of rule instances (derived from different rules and asserting the same conclusion) will be a function of the *lack or presence of positive/negative correlation* among the rules in each subset. The ordering of the T-norms reflects the transition from the case of extreme negative correlation or mutual exclusiveness (T_1), through independence (T_2), to the case of extreme positive correlation or subsumption (T_3).

In all the assignments, a set of selection rules will express the meta-knowledge about the context (i.e, the task's relevance to the decision maker and the subsets of deduction rules used to solve that task, see Section 3.). The selection rules will select the T-norms that better reflect the desired attitude toward risk and the perceived amount of correlation to be used in such a context.

2. Belief Revision

An initial implementation of the belief revision of uncertain information is available in the control layer of RUM's Rule System. For any conclusion made by a rule, the belief revision mechanism monitors the changes in the certainty measures of the *wffs* that constitute the conclusion's support and the changes in the calculus used to compute the cached conclusion certainty measure. Validity flags are inexpensively propagated through the rule deduction graph, which includes both wffs and rules. Five flag values are used: Good, guaranteeing validity; Bad (level i), indicating unreliability propagated to the ith level; Inconsistent, indicating conflict; Not Applicable, indicating that the rule context is not active; and Ignorant, indicating that the information is too vague to be useful.

The belief revision system offers both backward and forward processing. A *lazy evaluation*, running in backward mode, recomputes the certainty measures of the modified *wffs* that are required to answer a given query. This mode (called *reasoning under pressure*) is used when the system or the user decide that they are dealing with time-critical tasks. Breadth-first, forward mode processing recomputes the certainty measures of the modified *wffs*, attempting to restore the integrity of the rule deduction graph. This mode is used by the system when time is not critical.

3. Rule Firing Control via Context Activation

A rule context is defined as a set of conditions that must be satisfied before the rule can be considered for premise evaluation. A user-definable threshold can be attached to each rule context, either by local definition or by inheritance from a rule class. The semantics of a context C attached to an inference rule (establishing the weak logical equivalence between A and B) is given by the following expression:

$$C \rightarrow (A \xleftrightarrow{s,n} B)$$

It is important to note that the inference symbol $\rightarrow$ in the production rule $A \xrightarrow{s} B$ is interpreted as a *(weak) material*

implication operator in multiple-valued logics. The value s is the lower bound of the degree of sufficiency of the implication. This is different from the idea of *conditioning*, i.e., $s = P(B|A)$.

The symbol $\leftrightarrow$ in the production rule $A \overset{s,n}{\longleftrightarrow} B$ is interpreted as a *(weak) logical equivalence* operator in multiple-valued logics, where s and n are the lower bounds of sufficiency and necessity, respectively. This (weak) logical equivalence is an *if-and-only-if* rule that could be decomposed in the following two rules: $A \overset{s}{\rightarrow} B$ and $B \overset{n}{\rightarrow} A$ (equivalent to $\neg A \overset{n}{\rightarrow} \neg B$. RUM's rules are of the type $C \rightarrow (A \overset{s,n}{\longleftrightarrow} B)$, where C indicates the context of the rule and $\rightarrow$ represents a strong material implication.

The context mechanism provides the following features:

1. By activating/deactivating subsets of the KB, it limits the number of rules that will be considered relevant at any given time, thus increasing the overall system efficiency.
2. By only considering the rules relevant to a given situation, it allows the knowledge engineer to effectively use the necessary conditions in the rule's premise. It is now possible to distinguish between the failure of a necessary test (described in the premise) and the failure of the rule's applicability (traditionally described by other clauses in the same premise and now explicitly represented in the context).
3. By using predicates on the control-level wffs, it provides the required programmability for defining flexible control strategies, such as causing sequences of rules to be executed, firing default rules, ordering and handling time-dependent information, etc.
4. By using hierarchical contexts, an organizing principle is created for easing knowledge acquisition.

4. Rule Tracing

A user-definable threshold pair can be attached to each rule, either by local definition or by inheritance from a rule class. The threshold pair defines a lower and upper bound of certainty to be compared with the the certainty interval of the detached conclusion. If either threshold is exceeded, the firing mechanism will execute any user-defined LISP function attached to the rule, before the conclusion is detached. This property will enable the user to build any desired tracing facilities, or to create markers for use by the contexts of other rules.

IV. Testing RUM

RUM's capabilities were first tested in October 1986, as part of an experiment in simulated naval situation assessment. Figure 2 illustrates the configuration of the components used in such experiment.

A. An Object-based Simulation Environment

The simulation of the scenario was implemented in LOTTA, an object-oriented symbolic battle management simulator that maintains time-varying situations in a multi-player antagonistic game [Bonissone *et al.*, 1987]. A development environment centered around LOTTA was the testbed used to test the new techniques in reasoning with uncertainty. The development environment is composed of four basic modules: the *Window Manager*, a map-like window-oriented user interface; the *Annotation System*, an intelligent database for LOTTA; *LOTTA*, the simulator that executes commands and maintains internal states; and *KEELA*, (KEE to LottA interface) the link between LOTTA and RUM.

B. Information Fusion and Situation Assessment

The situation assessment problem is composed of several tasks in which uncertainty pervades both the input data and the knowledge bases. Given a platform (aircraft, ship, tank) in a potentially hostile environment, the process of situation assessment consists of the following tasks:

- Sensor data must be collected from various sources and described as *reports*
- Time-stamped sensor reports must be consolidated into *tracks* (each track is the trace of an object followed by a given sensor)
- Tracks associated to the same object must be fused into a *platform*
- The detected platform must be *classified* and *identified* (by class and type)
- Node organization (*formation* of the identified platforms), use of special equipment, and maneuvering must be recognized
- Using the knowledge of the opponent's doctrines and rules of engagement, the recognized formation and observed use of special equipment must be explained by a *probable intent*, which is then translated into a *threat assessment*.

The first four tasks constitute what is generally known as information fusion, and they define the scope of the simplified example described in the next section.

C. A Simplified Example in Information Fusion

In one experiment, a modified version of the naval situation assessment scenario used by NOSC to test STAMMER and STAMMER2 [McCall *et al.*, 1979, J.P. Ferranti, 1981] was created. In it, a single missile cruiser faced three platforms that were either patrol hydrofoils or merchant ships. The cruiser's task was to track, correlate, and classify each detected object. Both passive and active sensors on the cruiser were run twice, generating sensor reports which were grouped into tracks for each sensor. Plausible correlations were then made between tracks to group them into detected platforms. Using the RUM knowledge base (see the example in the next section) on this information, the platforms were correctly identified.

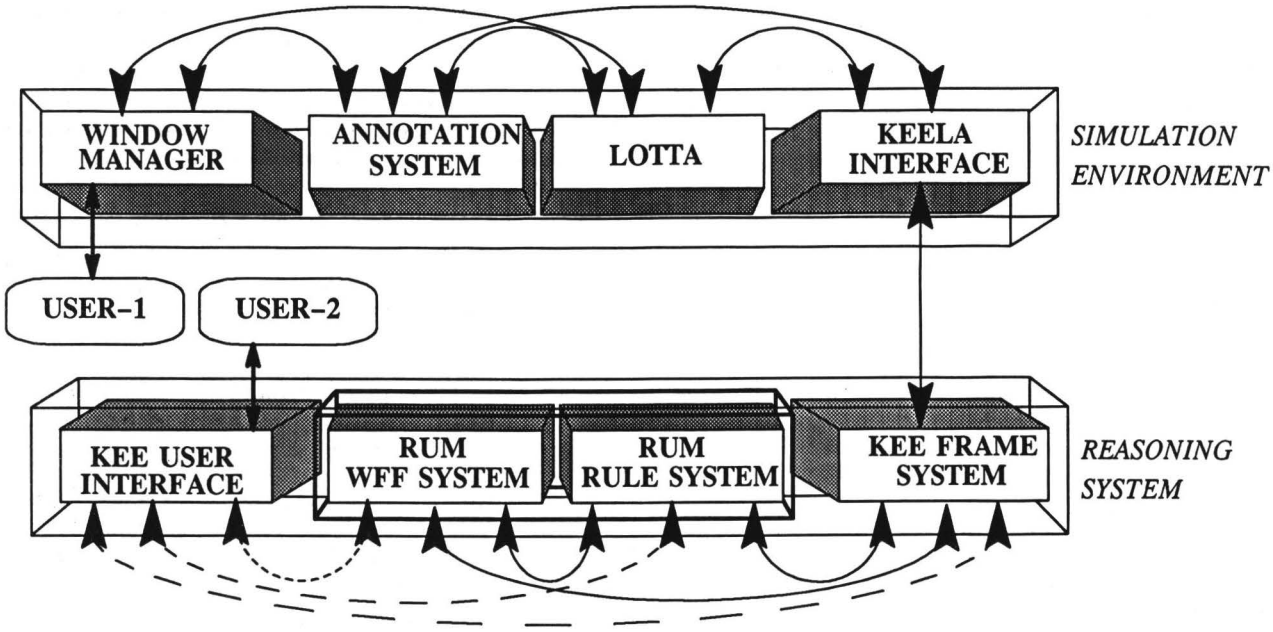

Figure 2: Architecture for Simulated Naval Situation Assessment

1. Example of a RUM rule

The RUM knowledge base (KB) used in this example is composed of approximately forty rules, each of which can be instantiated by new sensor reports, new tracks, or new platforms. A representative sample of such a KB is provided by the following rule.

English Version of Rule-550 (identifying submarines): *Assuming that sonar was used to generate a sensor report (which together with other reports generated by the same sensor has been attached to a track associated with a platform), if the detected platform has a low noise emission, and is located at a depth of at least twenty meters, then it is extremely likely that the detected platform is a submarine. Otherwise, the detected platform may not be a submarine.*

RUM's Version of the same rule:

```
(add-template 'sub.pos.id-low.noise-550                    ; Name
   'msmt                                                    ; KB
   '((is-value?  ?report 'noise-emissions 'low)    ; Premises
     (u-lessp (get.uncertain.value ?report 'elevation)
              (fuzz -20)))
   '(((get.platform ?report)
        class.name submarine s25.rules))      ; Consequences
   '((is-in-class? (get.value ?report 'track)
                   'source
                   '(sensor-sonar-mixin lotta)))    ; Context
   '(extremely.likely it.may)     ; Sufficiency and Necessity
   't3)                                  ; Aggregation T-norm
```

V. Remarks and Conclusions

We have implemented a layered architecture for reasoning with uncertainty. In the representation layer we have used frame-like data structures attached to each variable. Each frame contains a list of the values that were considered for the variable. For each of these values, RUM maintains uncertainty information, such as the lower and upper bounds, and uncertainty meta-information, such as a measure of ignorance, a measure of consistency, and the evidence source.

In the inference layer, we have used T-norm based calculi that are truth functional and associative. The truth functionality of the calculi entails low computational complexity and relatively inexpensive belief revision. The associativity of the calculi entails recursive problem decomposition. The formulae for detachment have been derived from multiple-valued logics, by interpreting the inference symbol of the production rules as material implication, rather than conditioning. Instead of adopting a unique calculus (based on global assumptions) and uniformly using it in the knowledge base, we can select any of the five T-norm based calculi and locally apply them to any given rule subset.

In the control layer, we have implemented the capabilities of selecting the appropriate calculus based on the calculi characteristics (context independent information) and on the available meta-information describing the context and the user's attitude toward risk (context dependent information). We have also implemented a belief revision mechanism that can be frugal for time-critical situations (operating in a depth-first backward mode) or exhaustive (operating in a breadth-first forward mode). Rule firing is controlled via a context activation mechanism that reduces the number of rules that will be considered relevant at any given time.

Future work on RUM will focus on extensions of the control layer capabilities. Specifically, we will define a meta-language for describing the policies (i.e., meta-rules) for cal-

culus selection, ignorance reduction, and conflict removal; we will also extend the meta-language to enable the use of the context mechanism from the meta-rules. Finally, we will start addressing real-time requirements by incorporating the timeliness of information as another constraint in the control of reasoning represented in the control layer.

References

[1986] *KEE Software Development System User's Manual.* Intellicorp, Mountain View, CA. 94040–2216, KEE Version 3.0 edition, July 1986. Document Number 3.0–U–1.

[Beyth-Marom, 1982] R. Beyth-Marom. How probable is probable? a numerical taxonomy translation of verbal probability expressions. *Journal of Forecasting*, 1:257–269, 1982.

[Bonissone, 1982] P.P. Bonissone. A fuzzy sets based linguistic approach: theory and applications. In M.M. Gupta and E. Sanchez, editors, *Approximate Reasoning in Decision Analysis*, pages 329–339, North Holland, New York, 1982.

[Bonissone, 1986] P.P. Bonissone. Summarizing and propagating uncertain information with triangular norms. In L. S. Baumann, editor, *Proceedings of the Expert Systems Workshop*, pages 62–71, Defense Advanced Research Projects Agency, April 1986. To appear in the International Journal of Approximate Reasoning, North-Holland, 1987.

[Bonissone and Decker, 1986] P.P. Bonissone and K.S. Decker. Selecting uncertainty calculi and granularity: an experiment in trading-off precision and complexity. In L. N. Karnak and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, North Holland, 1986. Also GE Technical Report 85-CRD-171.

[Bonissone *et al.*, 1987] P.P. Bonissone, J.K. Aragones, and K.S. Decker. Lotta: an object based simulator for reasoning in antagonistic situations. 1987. GE Corporate Research & Development Working Paper.

[Bonissone, 1987] P. P. Bonissone. Plausible reasoning: coping with uncertainty in expert systems. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, John Wiley and Sons, 1987. Also GE Technical Report 86-CRD-53.

[Cohen and Grinberg, 1983a] P.R. Cohen and M.R. Grinberg. A framework for heuristics reasoning about uncertainty. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 355–357, Karlsruhe, West Germany, 1983.

[Cohen and Grinberg, 1983b] P.R. Cohen and M.R. Grinberg. A theory of heuristics reasoning about uncertainty. *AI Magazine*, 4(2):17–23, 1983.

[Doyle, 1983] J. Doyle. Methodological simplicity in expert system construction: the case of judgements and reasoned assumptions. *AI Magazine*, 4(2):39–43, Summer 1983.

[Genesereth, 1982] M.R. Genesereth. *An Overview of MRS for AI Experts.* memo HPP-82-27, Stanford Heuristic Programming Project, Stanford University, 1982.

[J.P. Ferranti, 1981] Jr. J.P. Ferranti. *Evaluation of the artificial intelligence program STAMMER2 in the tactical situation assessment problem.* Technical Report #AD-A101101, National Technical Information Service, March 1981.

[McCall *et al.*, 1979] D.C. McCall, P.H. Morris, D.F. Kibler, and R.J. Bechtel. *STAMMER2 production system for tactical situation assessment.* Technical Report #AD-A084038, National Technical Information Service, October 1979.

[Schweizer and Sklar, 1963] B. Schweizer and A. Sklar. Associative functions and abstract semi-groups. *Publicationes Mathematicae Debrecen*, 10:69–81, 1963.

[Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, New Jersey, 1976.