PLASMA: Combining Predicate Logic and Probability for Information Fusion and Decision Support

Francis Fung[1], Kathryn Laskey[2], Mike Pool[1], Masami Takikawa[1], Ed Wright[1]

[1]Information Extraction and Transport, Inc.
1911 North Fort Myer Dr., Suite 600
Arlington, VA 22209
[2]Department of Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030
{fung, pool, takikawa, wright}@iet.com
klaskey@gmu.edu

Abstract

Decision support and information fusion in complex domains requires reasoning about inherently uncertain properties of and relationships among varied and often unknown number of entities interacting in differing and often unspecified ways. Tractable probabilistic reasoning about such complex situations requires combining efficient inference with logical reasoning about which variables to include in a model and what the appropriate probability distributions are. This paper describes the PLASMA architecture for predicate logic based assembly of situation-specific probabilistic models. PLASMA maintains a declarative representation of a decision theoretically coherent first-order probabilistic domain theory. As evidence about a situation is absorbed and queries are processed, PLASMA uses logical inference to reason about which known and/or hypothetical entities to represent explicitly in the situation model, which known and/or uncertain relationships to represent, what functional forms and parameters to specify for the local distributions, and which exact or approximate inference and/or optimization techniques to apply. We report on a prototype implementation of the PLASMA architecture within IET's Quiddity*Suite, a knowledge-based probabilistic reasoning toolkit. Examples from our application experience are discussed.

Introduction

Decision support in complex, dynamic, uncertain environments requires support for situation assessment. In an open world, situation assessment involves reasoning about an unbounded number of entities of different types interacting with each other in varied ways, giving rise to observable indicators that are ambiguously associated with the domain entities generating them. There are strong arguments for probability as the logic of choice for reasoning under uncertainty (e.g., Jaynes, 2003). Graphical models, and in particular Bayesian networks (Pearl, 1988), provide a parsimonious language for expressing joint probability distributions over large numbers of interrelated random variables. Efficient algorithms are available for updating probabilities in graphical models given evidence about some of the random variables. For this reason, Bayesian networks have been widely and successfully applied to situation assessment (Laskey et. al., 2000; Das, 2000).

Despite these successes, there are major limitations to the applicability of standard Bayesian networks for situation assessment. In open-world problems, it is impossible to specify in advance a fixed set of random variables and a fixed dependency structure that is adequate for the range of problems a decision support system is likely to encounter. Nevertheless, graphical models provide a powerful language for parsimonious specification of recurring patterns in the structural features and interrelationships among domain entities. In recent years, a number of languages have emerged extending the expressiveness of Bayesian networks to represent recurring structural and relational patterns as objects or frames with attached probabilistic information (Laskey and Mahoney, 1997; Koller and Pfeffer, 1997; Bangsø and Wuillemin, 2000). There is an emerging consensus around certain fundamental approaches to representing uncertain information about the attributes, behavior, and interrelationships of structured entities (Heckerman, et al., 2004). This consensus reflects fundamental logical notions that cut across surface syntactic differences. The newly emerging languages express probabilistic information as collections of graphical model fragments, each of which represents a related collection of generalizations about the domain. These model fragments serve as templates that can be instantiated any number of times and combined to form arbitrarily complex models of a domain. Recent work in probabilistic logic has clarified the semantics of such languages (e.g., Laskey, 2004; Sato, 1998; Bacchus, et al., 1997).

PLASMA has its logical basis in Multi-Entity Bayesian Network (MEBN) logic. A theory in MEBN logic consists of a set of Bayesian network fragments (MEBN fragments, or MFrags) that collectively express a joint probability distribution over a variable, and possibly unbounded number of random variables. MEBN logic can express a probability distribution over interpretations of any finitely axiomatizable theory in first-order logic, and can represent the accrual of new information via Bayesian conditioning (Laskey, 2004).

The PLASMA architecture is depicted in Figure 1. A domain model consisting of a set of MFrags represents probabilistic knowledge about types of entities (e.g., objects, processes, concepts), the attributes of each type, and the relationships in which they can participate. A first-order logic knowledge base expresses knowledge about how to combine the MFrags into a Bayesian network for reasoning about a given situation, where a situation is defined as a given collection of background information, reports from external sensors, and queries posed by the user. These Bayesian network construction rules are called *suggestors* because they encode suggested actions for the SSBN construction module to take. A data interchange module accepts queries from users, reports from sensors, and computational results from special-purpose external reasoners (e.g., a tracking, clustering, linear programming, or other special-purpose module). The data interchange model can also task sensors, request computations to be performed by external special-purpose reasoners, and provide situation updates to users. The SSBN construction module contains a logical reasoner that has access both to the suggestors from the FOL knowledge base and to the situation-specific probabilistic model. It performs FOL inference by querying the suggestor rules to derive a set of model construction actions. These actions may include retrieving and instantiating MFrags from the MFrag knowledge base; connecting the MFrag instances to the SSBN by adding arcs and/or possible values for random variables; and possibly pruning parts of the SSBN that have become unnecessary to represent explicitly.

The remainder of the paper is organized as follows. The next section discusses why integrating probability and logic is both important and difficult, and provides an argument for the approach taken by PLASMA. Next, an example is presented of the kind of problem to which PLASMA can be applied. The example is followed by a brief overview of MEBN logic and situation-specific Bayesian network construction. Next, the PLASMA architecture is described in greater detail, followed by conclusions and discussion.

Why Logic and Probability

An important technical challenge for combining first-order logic with probability is that most statements about an uncertain world are not known to be true, but rather are assigned a probability. It is known that attempts to apply first-order logic directly to uncertain situations can quickly lead to undesirable deductions within the model. Bamber (1998) gives an example in which a pure logical reasoner applies the two rules: (1) "Aircraft carriers can launch planes," and (2) "Aircraft carriers with broken catapults cannot launch planes," and draws the conclusion that an aircraft carrier with a broken catapult is a destroyer. This happens because in classical first-order logic, rules cannot have exceptions. If rules can have no exceptions, then there can be no aircraft carriers with broken catapults and any inference whatsoever about a null set of entities is vacuously true.

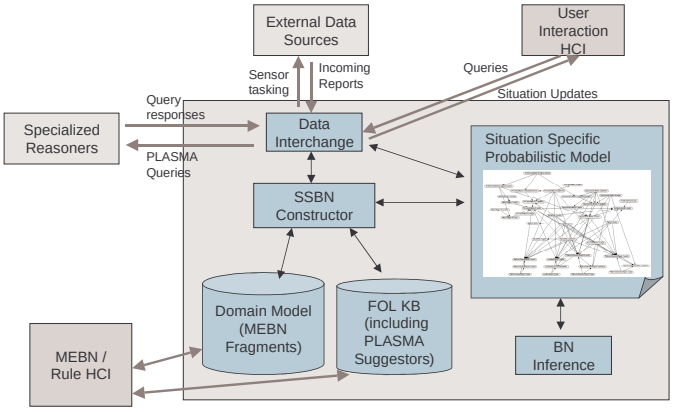

Figure 1: PLASMA Architecture

Of course, rules in the real world have exceptions. A burgeoning literature on default reasoning systems has arisen to cope with the problem of handling exceptions. However, there is no consensus on the fundamental principles that should govern the choice of when to apply defaults and what conclusions to retract when default conclusions conflict. This lack of a firm theoretical foundation gives rise to major practical difficulties for the application developer. When a system performs incorrectly, it is not clear whether the problem resides in the knowledge encoded in the system or in the heuristics built into the default reasoner. Additional difficulties arise from attempts to share knowledge between applications built under different reasoning paradigms. Heuristics applied by modelers to induce a default reasoner to behave well on a given application may work poorly when the knowledge is ported to a different default reasoner. These are the kinds of issues that led Bamber and others to argue for probability as a logical basis for design of rule-based reasoners. A probabilistic logic that can express probability distributions over interpretations of arbitrary first-order theories provides a solid foundation for a probabilistic semantics of defaults.

It is often natural to state relationships using logical implications even when the relationship is not strictly a logical implication. For example, consider the statement, "If a hostile ship is on a direct approach toward a friendly ship and is moving rapidly, then it is attacking." There is a considerable degree of fuzziness about the definition of terms such as "direct approach," or "moving rapidly," and the rule may be applied even when there is some question about whether there actually is an approaching ship or

whether the report is spurious. We would like to allow the modeler to express model construction actions in a natural rule form, without forcing the modeler to explicitly account for probabilistic relationships in the suggestor itself.

We have touched on some of the many difficulties that can arise from the attempt to give ad-hoc probabilistic interpretations to logical rules. IET's approach to overcoming this technical challenge is: *to maintain a fully probabilistic representation of the situation-specific model, to use logical inference only as a guide for constructing and augmenting the model, and then to compute query responses by performing probabilistic inference on the constructed model*. To achieve this, PLASMA provides a mechanism for promoting likely statements into provisional assumptions that are analyzed by a logical inference engine as if they were true. The inference engine is used to evaluate suggestor triggers that guide the construction of a situation-specific Bayesian network. That is, logical reasoning with default assumptions is used to select hypotheses that are sufficiently promising to merit explicit reasoning about their relative likelihoods, and exact or approximate probabilistic inference is used to evaluate these likelihoods.

For example, suppose that a limited amount of flu vaccine exists and we require judicious distribution in order to prevent localized outbreaks from turning into national epidemics. Suppose that we define a local outbreak as any situation in which more than ten people in a neighborhood have the flu. Suppose that our data are at the symptom level and that we also have data about patient addresses. Our reasoning tools must be able to adroitly move back and forth between probabilistically diagnosing individual cases based on symptoms, aggregating the individual cases, reasoning about which likely flu holders live in the same neighborhood, and evaluating the likelihood of outbreaks. PLASMA accomplishes this via threshold values on model construction operations. So, for example, we might set a threshold of .8 indicating that for purposes of our suggestor reasoning tools, any person whose likelihood of having the flu is greater than .8 is simply assumed to have the flu. These individuals are regarded as definite flu cases for purposes of constructing the situation-specific model, but when the model is applied to draw inferences, their medical conditions are again treated as uncertain.

A Case Study

We illustrate the PLASMA functionality using a scenario taken from Cohen, Freeman and Wolf (1996). The article presents a model of human cognition that the authors argue accounts for the decision making process of the U.S. captain and Tactical Air Officer (TAO). The authors state that the officers' reasoning was not Bayesian. We argue that although they are correct that this kind of situated, adaptive reasoning cannot be captured by a standard Bayesian network, it can be modeled quite naturally by MEBN logic and situation-specific Bayesian network construction.

As the scenario begins, U.S. ships were conducting freedom of navigation demonstrations in Libyan-claimed waters. The Libyan leader was threatening to attack any vessel that ventured below the "Line of Death." One night, a gunboat emerged from a Libyan port, turned toward an AEGIS-equipped cruiser, and increased its speed. The U.S. officers believed the gunboat's behavior indicated an intent to attack. First, a direct approach at a speed of 75 knots by a combat vessel from a hostile power is strongly indicative of an attack. Second, the Aegis cruiser was a logical target for an attack by Libya, being 20 miles within the "Line of Death." Furthermore, apparent missile launches toward other American ships had been detected earlier that day, indicating that Libya was actively engaging surface vessels.

However, there were conflicting indicators. The gunboat was ill-equipped to take on the Aegis cruiser, let alone the larger U.S. fleet poised to come to its aid. The Libyans possessed air assets that were far better suited for such an assault. However, the officers reasoned that the Libyans might be willing to use every asset at hand against the U.S., regardless of the consequences to the striking asset. On the other hand, a more natural target for attack would have been another U.S. cruiser even further below the "Line of Death."

A more serious complication was that the officers believed that the gunboat probably did not have the capability to detect the cruiser at the range at which it had turned inbound. Perhaps, they reasoned, the gunboat was receiving localization data from another source, or perhaps it possessed superior sensing technology of which U.S. intelligence was unaware. On the other hand, virtually any maneuver by the track would have put it on a vector to some friendly ship. Perhaps its direct line of approach was merely a coincidence.

Thus, the evidence regarding the gunboat's intent was ambiguous and conflicting. The officers considered several hypotheses, juxtaposing each against the available background information and situation-specific evidence, in an attempt to ascertain the intent of the gunboat. They considered the hypothesis that the gunboat was on patrol, but discarded that hypothesis because its speed was too fast for a ship on patrol. The hypothesis they finally settled on was that the gunboat was following a plan of opportunistic attack (i.e., to proceed into the Gulf ready to engage any ship it encountered). This hypothesis explained its rapid speed, its inability to localize the Aegis cruiser, and the fact that it was not the most appropriate asset to target against a U.S. Aegis cruiser.

We developed a knowledge base consisting of:

- A set of entity types that represents generic knowledge about naval conflict. These include:
  - *Combatants*: Friendly and hostile combatant MFrags represent background knowledge about their high-level goals and their level of aggressiveness.
  - *Plans*: There are plans for provoked attacks, opportunistic attacks, and patrols, as well as a generic non-specific plan type to represent a plan

type not included among the hypotheses being considered.
    - *Attack Triggers*: An attack trigger is a kind of behavior that might provoke an attack. We modeled the U.S. presence below the "Line of Death" as an instance of an attack trigger.
    - *Reports*: Reports are used to model observable evidence that bears on the hypotheses under consideration.
- A set of suggestors that represents knowledge about which hypotheses should be explicitly considered in given kinds of situation. Suggestors are expressed as first-order rules. The rules operate on observed evidence (e.g., the cruiser is situated below the "Line of Death"; the gunboat is on a rapid direct approach), together with information about the current status of the Bayesian network (e.g., no attack hypothesis has yet been enumerated for the gunboat), and nominate suggested network construction actions. Future implementations will allow suggestors to nominate pruning actions that delete low-probability hypotheses from the probabilistic KB, to help make inference more efficient.
- Particular facts germane to the scenario, including background knowledge and observed evidence.

The suggestors for the gunboat model were based on the following general rules, which are reflective of the kinds of reasoning the officers applied in this scenario.

- Any ship sailing in its own territorial waters may be on patrol. This was implemented as a suggestor nominating a patrol as a possible plan for any hostile ship.
- If a friendly ship is engaging in behavior against which enemy has threatened to retaliate, then an attack may occur. In this particular scenario, venturing within the Line of Death might provoke an attack.
- A hostile asset approaching on a bearing directly toward a ship may be attacking the ship it is approaching. A rapidly approach strengthens this inference. These rules were not implemented as suggestors, but were instead represented as likelihood information for the attack plans.
- In a provoked attack, the attacker must be able to localize the target it is attacking in advance of the attack. In an opportunistic attack, this is unnecessary. This knowledge was implemented as likelihoods on localization given provoked and opportunistic attacks. There is also a suggestor that fires when the probability of an attack exceeds a threshold, that generates a request for evidence on whether the attacker can localize the target, and another suggestor that inserts the result of the report if the report exists.
- When there is evidence consistent with an attack but the evidence conflicts with the hypothesis of a deliberate, provoked attack, then opportunistic attack is a possible explanation. This rule was implemented as follows. There was a suggestor nominating a generic "other" plan for any asset, representing a plan other than the ones explicitly enumerated. There was another suggestor specifying that if an attack has been hypothesized, but the probability of "other" exceeds a threshold (an indication of conflicting evidence), then hypothesize an opportunistic attack as a possible plan for the asset.

We exercised PLASMA on this scenario, obtaining qualitative results similar to the analysis reported by Cohen, et al. Initially, the provoked attack hypothesis dominates, but as the evidence is processed, its incongruity with the provoked attack hypothesis becomes more apparent. This increases the probability of the "other" hypothesis. A natural hypothesis to consider is the patrol hypothesis, but it too is incongruent with the available evidence. When the opportunistic attack hypothesis is nominated in response to the failure of other plans to account for the evidence, it becomes the dominant hypothesis.

MEBN Logic

MEBN logic combines the expressive power of first-order predicate calculus with a sound and logically consistent treatment of uncertainty. MEBN fragments (MFrags) use directed graphs to specify local dependencies among a collection of related hypotheses. MTheories, or collections of MFrags that satisfy global consistency constraints, implicitly specify joint probability distributions over unbounded and possibly infinite numbers of hypotheses. MEBN logic can be used to reason consistently about complex expressions involving nested function application, arbitrary logical formulas, and quantification. A set of built-in MFrags gives MEBN logic the expressive power of first-order logic.

A set of MFrags representing knowledge for modeling a gunboat attack scenario is shown as Figure 2. Each MFrag contains a set of *random variables*. Random variables are functions that map the entities filling their arguments to a set of possible values characteristic of the random variable. For example, *IntendedOutcome(p)* maps a plan *p* to a value representing the outcome the agent intends to achieve. Each MFrag contains *resident* random variables, whose distribution is defined in the MFrag, *input* random variables, whose distribution is defined in another MFrag, and context random variables, which represent conditions under which the distribution defined in the MFrag applies. Whether a random variable is resident, input or context is indicated by shading and bordering in Figure 2: bold-bordered and dark gray indicates context, light gray and thin-bordered indicates input, and white indicates resident. Our example also includes report MFrags which are not shown in the figure for reasons of space.

Logical and mathematical relationships can also be expressed as MFrags. For example, in the gunboat scenario the officers received a report that the gunboat was headed directly toward the U.S. cruiser. For purposes of this illustrative example, we chose to represent the direction an asset was traveling with respect to its target as a random variable with four possible values:

DirectApproach, *GenerallyToward*, *NotToward*, and *NotRelevant*, and we assumed that evidence was provided as likelihoods for these values. In a fielded decision support application, a sensor would provide likelihood information about the heading of the gunboat and the geospatial coordinates of the U.S. ship. A coordinate transformation would be applied to obtain the bearing of the gunboat with respect to the U.S. ship. Although an implementation might choose for efficiency reasons to have coordinate transformations performed externally, MEBN logic is capable of representing such transformations as MFrags.

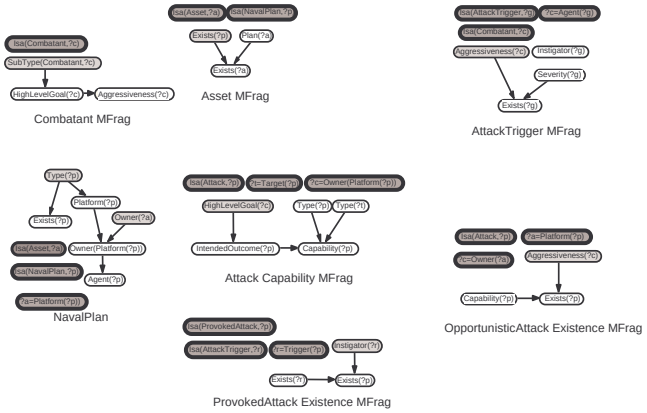

Figure 2: MFrags for Naval Attack MTheory

Typically, context random variables represent type and relational constraints that must be satisfied for the MFrag to be meaningful. A *default distribution* assigns a probability distribution when the context constraints are not met. In some cases, the default distribution will assign probability 1 to the undefined symbol ⊥; in other cases, it is appropriate to assign a "leak" distribution representing the influence of not-yet-enumerated parents.

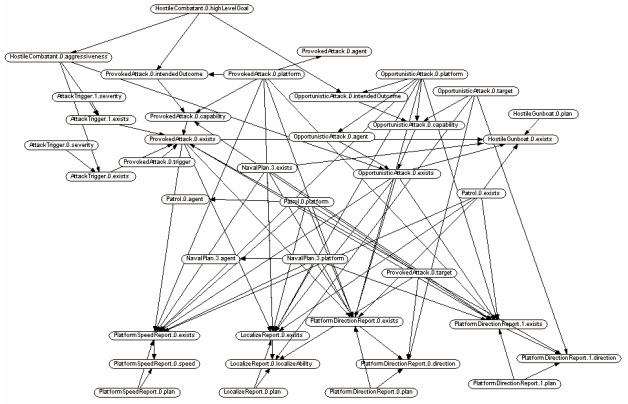

Figure 3: Situation-Specific Bayesian Network

The MFrags of Figure 2 define a coherent (that is, consistent with the constraints of probability theory) probability distribution over entities in the domain. This probability distribution can be used to reason about group membership of different numbers of objects reported on by sensors with varying spatial layouts. For instance, Figure 3 shows a situation-specific Bayesian network (SSBN) for a situation with two friendly cruisers and two enemy assets, and several attack hypotheses. This SSBN was generated by instantiating and assembling MFrags that represent the relevant entities.

A straightforward combinatorial argument shows, however, that this extremely simple MTheory can result in situation-specific models of staggering computational complexity when applied to problems with many assets and significant uncertainty about the plans the assets are pursuing. In typical problems encountered in applications, excellent results can be obtained by applying heuristic model construction rules to identify and include those random variables most relevant to a given query, and to exclude those having sufficiently weak influence to be ignored.

PLASMA Architecture

To address the challenge of open-world situation assessment, IET has developed, as part of its Quiddity*Suite modeling and inference environment, an Execution Management framework (Quiddity*XM) for specifying and constructing situation-specific probabilistic models. Situation-specific models are assembled from domain models expressed in the Quiddity*Modeler probabilistic frame and rule language. A Quiddity*Modeler frame represents a type of entity, its attributes (slots in the frame), and the relationships it can bear to other entities (reference slots that can be filled by instances of frames). Each frame encodes an MFrag, where the resident random variables correspond to slots for which the frame defines a distribution, the input random variables correspond to slots in other frames that influence the distributions of resident random variables, and the context random variables correspond to logical conditions under which the distributions are well-defined. A domain model also includes *suggestors*, which are rules that represent knowledge about how to construct situation-specific models. Suggestors can specify which frames to instantiate or modify in a given situation, and how to connect the frame instances to existing portions of the model. The execution manager uses suggestors to manage the size and complexity of the situation-specific network. Quiddity*XM draws from the data-driven and query-driven paradigms (Wellman, et al., 1992) of knowledge-based model construction to assemble situation-specific models in response to both incoming data and queries.

As the suggestors recognize trigger conditions, they make suggestions about which kind of objects or events exist or might exist in the domain, and the kinds of relationships these objects bear to the previously observed or posited objects. In the original Quiddity*XM, suggestors were implemented as small, often stateless, procedural code modules. This approach is powerful, and provides all the computational advantages of procedural code versus declarative languages. However, the need to write programs to express model construction knowledge puts an undue burden on modelers to perform outside their area of primary expertise. IET has attempted to make

suggestors less opaque with respect to our representation language by moving them into the declarative knowledge-based reasoning system within which our modeling language is embedded. Thus, suggestors are represented as first-order rules encoding knowledge about which hypotheses to enumerate and which relationships to represent, as a function of current beliefs, current evidence, and the current query. The PLASMA architecture allows the reasoner to move gracefully back and forth between conclusions reached in the probabilistic reasoning system and the first order logical reasoning system. Logical reasoning about model construction actions is interleaved with probabilistic inference within the current model.

We illustrate with our Libyan gunboat example how PLASMA can implement first order-logic to represent knowledge that facilitates the efficient creation of situation specific networks. The DSS knowledge base contains a domain model consisting of modeler-specified MFrags and SSBN construction rules. These MFrags are represented as frames with uncertain slots annotated with probability information. In this problem, there are frames representing combatants, naval assets, plans, reports, and other entities of interest. These MFrags can represent relationships between entities of different types (e.g., the owner slot of a naval asset instance is filled by a combatant instance; the target slot of a hostile attack plan instance is filled by a friendly naval asset instance). Along with the MFrags, the modeler also creates a set of suggestors that describe conditions under which the situation-specific probabilistic model should be revised, and network construction actions to be taken under the given conditions. These suggestors are represented as Prolog-style rules that suggest conjunctions of network construction actions to be executed as a bundle.

At run-time, this knowledge base is exercised against reports and user queries to construct a situation-specific Bayesian network that represents query-relevant information about the current situation. At any given time, there will be a current situation-specific Bayesian network that reflects currently available reports and their implications for queries of current interest. In our example, the query of interest is the gunboat's plan. Given this current situation-specific Bayesian network, the Bayesian inference module draws conclusions about the likelihood of the different possible plans for the gunboat.

Changes to the current SSBN can occur as a result of incoming reports, queries posed by the user, or internal triggers stated in terms of the current belief state. The current prototype uses a Prolog-type language and inference, reasoning with Horn clauses. The PLASMA model construction paradigm can be described as follows. First, there is the FOL KB, where the modeler defines the vocabulary to express trigger conditions. Using the supplied interface predicates for interfacing with the probabilistic KB, the modeler can capture a fine-grained, expressive set of assertions that declare when particular model construction actions should take place. For instance, we can define a trigger that asserts that "IF ship is slightly below LOD, THEN moderately severe attack trigger should be hypothesized":

```
instigatesAttackTrigger(?shp,Libya,Moderate) :-
    location(?shp, BelowLOD, Slightly);
```

Then, the modeler declares what actions should be taken as a result of the appropriate trigger conditions being satisfied. These are declared in a convenient form

`<predicateName>`(<*set of actions to be taken*>) :-
<*trigger conditions*>.

For instance, the following suggestor indicates that a hypothetical instance of `AttackTrigger` (i.e., one with an uncertain probability of existing) should be declared if the trigger condition holds.

```
hypotheticalEntity(AttackTrigger,"instigator",
                                    ?platform):-
   instigatesAttackTrigger(?platform,?agnt,
                                    ?severity)
```

The `hypotheticalEntity` predicate is one of a set of generic PLASMA predicates which modelers can use in the construction of domain-specific suggestors. This predicate is used to suggest the model construction action of creating a hypothetical instance of the `AttackTrigger` frame and setting its "`instigator`" slot to the frame instance bound to `?platform`. Specifically, when a binding succeeds for `hypotheticalEntity` such that its first argument is a frame type, its second argument is a reference slot for the frame type, and its third argument is an instance of the correct type for the slot, the following network constructions are suggested:

- Create a hypothetical instance of the frame type (`AttackTrigger` in this example);
- Set the third argument as a possible value of the slot denoted by the second argument (in this example, the U.S. cruiser, which binds to `?platform`, becomes a possible value for the "instigator" slot of the `AttackTrigger` frame instance).

A hypothetical instance of a frame corresponds to a possible entity of the given type that may or may not be an actual entity. Each hypothetical entity has an "`exists`" slot, with a probability distribution representing the likelihood that the entity actually exists. In this case, the attack trigger is hypothetical because it is not known whether presence below the Line of Death actually will trigger an attack.

As reports arrive, new nodes are added to the SSBN to represent the reports; the reports are connected to entities (platform and plan instances) that might explain them; beliefs are updated to reflect the new information; and new hypotheses are introduced when warranted. The incremental model revision process is controlled by the built-in and modeler-defined suggestors.

Table 1 shows a few of the generic suggestors in the PLASMA library. The generic suggestors are available for use in constructing domain-specific suggestors. The table also gives examples of how these generic suggestors are applied in the gunboat scenario.

Declarative specification of suggestors makes our knowledge representation language more robust and

scalable than procedural encoding, as facilitates the explicit representation of semantic information that can be applied flexibly in a variety of contexts. The suggestors applied in the gunboat scenario could of course be implemented as procedural code. However, suggestors often draw upon semantic knowledge such properties of sets, partial orderings, or spatio-temporal phenomena. A declarative representation allows suggestors to draw upon upper ontologies for set-theoretic, mereological, or spatio-temporal reasoning, to compose these concepts as necessary for each new suggestor, and to rely on reasoning services optimized to a given ontology when executing suggestors related to that ontology. Also, note that the recognition of trigger condition A may require recognition of trigger condition B which in turn requires recognition of trigger condition C, etc. Backward chaining can be used to implement such reasoning chains.

Suggestor	Example Usage
Instantiate known entities & set known properties	A hostile gunboat is approaching Libya's high-level goal is protecting territorial rights
Hypothesize new entities to fill unfilled roles	"Unknown plan" hypothesis should be introduced for any entity with no enumerated plan
Represent reciprocal & composed relationship hypotheses	If `HostileGunboat.0` may be the platform performing `ProvokedAttack.1`, then `ProvokedAttack.1` may be the plan of `HostileGunboat.0`
Specify probability thresholds for model construction actions	If the probability of an attack exceeds 0.6 then call for a report on whether attacker can localize target

Table 1: Some PLASMA Generic Suggestors

Considerations such as these suggest that a modeler-friendly toolkit must be able to represent general concepts such as transitive reasoning, set membership, existential quantification, and spatio-temporal reasoning. Furthermore, the toolkit must provide logical reasoning services such as subsumption, constraint satisfaction, and forward and backward chaining in a way that hides implementation details from modelers. Such a toolkit enables a scaleable and robust approach to suggestor development, allowing modelers to use existing atomic concepts to articulate new complex trigger conditions for suggestors. Hence, we have adopted first order logic as the basis for our suggestor language; we have implemented a Prolog-like backward chaining algorithm for basic logical reasoning; and our architecture allows an application to interface with special-purpose reasoners when appropriate.

The PLASMA suggestor language provides the ability to specify declaratively the conditions under which a given MFrag or collection of MFrags should be retrieved and instantiated. Such conditions include satisfaction of the context constraints of the MFrags, as well as additional constraints on the input and resident random variables of the MFrag. These additional constraints typically involve conditions on the marginal likelihood of input random variables which, when satisfied, tend to indicate that the relevant hypotheses are sufficiently likely to merit consideration. Logical reasoning is then applied to identify promising random variables to include in the situation-specific model. Suggestors can also provide numerical scores used to prioritize model construction actions and control rule execution. The resulting architecture generalizes logical inference to uncertain situations in a mathematically sound manner, while freeing the modeler from tedious and error-prone code development.

Once a situation specific model has been constructed, we also require the ability to prune hypotheses that are either extremely unlikely or very weakly related to the results of a query of interest. Logical reasoning can also be used to apply thresholds or other rules for pruning parts of a model that are only weakly related to the desired query result.

Many of our suggestors are strictly heuristic, but we have also developed suggestors based on decision theoretic reasoning applied to an approximate model. For example, if we specify penalties for false alarms, missed identifications, and use of computational resources, then an MTheory can represent a decision model for the task model construction problem, in which computational cost is traded off against task performance. Of course, exact solution of this decision model may be more computationally challenging than the task model itself, but we may be able to specify a low-resolution approximation that is accurate enough to provide a basis for suggestor design. We can allow modelers to specify a suggestor as a fragment of an influence diagram, and then automatically translate the decision regions into logical rules to be applied by the logical reasoning engine. Collections of suggestors can be evaluated and compared by running simulations across a variety of scenarios.

The PLASMA system realizes an efficient division of labor by exploiting the respective strengths of logic-based and probability-based representation and reasoning paradigms. We use a logic-based tool for managing the inventory of objects in our reasoning domain. Variable binding, transitive reasoning, set theoretic reasoning and existential quantification is available for purposes of maintaining and pruning our inventory of objects in our domain, recognizing relevant relationships and quickly connecting relevant objects for purposes of representing and reasoning about when and how to assemble our more fine-tuned Bayesian representation tools. However, we are also able to quickly instantiate a more focused, fine-grained probabilistic model for the complex situational reasoning that the Bayesian tools can then perform. A somewhat useful analogy here is the surgeon and surgeon's assistant. The assistant keeps the workspace clean and uncluttered ensuring that the right tools are available for the surgeon's effort. Similarly, we have implemented the PLASMA suggestor logic tool as a "surgeon's assistant" ensuring that the precise tools of our Bayesian reasoner are not hindered by being forced to

consider irrelevant objects and relationships, but can be focused on the accurate fine-grained reasoning that it does best.

Summary and Conclusions

The PLASMA architecture combines logical and probabilistic inference to construct tractable situation-specific Bayesian network models from a knowledge base of model fragments and suggestor rules. This architecture permits us to smoothly integrate the representational advantages of two distinct reasoning and representational paradigms to overcome central challenges in hypothesis management. Logic-based concept definitions encode complex definitional knowledge about the structure and makeup of objects in the reasoning domain while probabilistic knowledge allows us to process uncertain and potentially contradictory information about the actual objects under observation.

This architecture permits declarative specification of model construction knowledge as well as knowledge about the structure of and relationships among entities in the application domain. The concepts underlying PLASMA have been applied to problems in several application domains such as military situation assessment and cyber security. We believe that this prototype implementation will increase the accessibility of our modeling tools, and provide reasoning services that take care of implementation details, thus freeing modelers to concentrate on how best to represent knowledge about the domain.

Acknowledgements

This research has been supported under a contract with the Office of Naval Research, number N00014-04-M-0277. We are grateful to Suzanne Mahoney and Tod Levitt for their contributions to the ideas underlying this work.

References

Bacchus, F., Grove, A., Halpern, J. and Koller, D. 1997. From statistical knowledge bases to degrees of belief. *Artificial Intelligence* **87**(1-2):75-143.

Bamber, D. 1998. How Probability Theory Can Help Us Design Rule-Based Systems. In Proceedings of the 1998 Command and Control Research and Technology Symposium, Naval Postgraduate School, Monterey, CA, pp. 441–451.

Olav Bangsø, O., and Wuillemin, P.-H. 2000. Object Oriented Bayesian Networks: A Framework for Topdown Specification of Large Bayesian Networks and Repetitive Structures, Technical Report CIT-87.2-00-obphw1, Department of Computer Science, Aalborg University.

Cohen, M., Freeman, J., and Wolf, S. 1996. Metarecognition In Time-Stressed Decision-Making. *Human Factors* **38**(2):206-219.

Das, B. 2000. Representing Uncertainties Using Bayesian Networks, Technical Report, DSTO-TR-0918, Defence Science and Technology Organization, Electronics and Surveillance Research Lab, Salisbury, Australia.

Heckerman, D., Meek, C. and Koller, D. 2004. Probabilistic Models for Relational Data, Technical Report MSR-TR-2004-30, Microsoft Corporation.

IET, Quiddity*Suite Technical Guide. 2004. Technical Report, Information Extraction and Transport, Inc.

Jaynes, E. T. 2003. *Probability Theory: The Logic of Science*. New York, NY: Cambridge Univ. Press.

Koller, D. and Pfeffer, A. 1997. Object-Oriented Bayesian Networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference*, 302-313. San Mateo, CA: Morgan Kaufmann Publishers.

Laskey, K. B. 2004. MEBN: A Logic for Open-World Probabilistic Reasoning, Working Paper, http://ite.gmu.edu /~klaskey/papers/Laskey_MEBN_Logic.pdf, Department of Systems Engineering and Operations Research, George Mason University,

Laskey, K.B., d'Ambrosio, B., Levitt, T. and Mahoney, S. 2000. Limited Rationality in Action: Decision Support for Military Situation Assessment. *Minds and Machines*, **10**(1), 53-77.

Laskey, K. B. and Mahoney, S. M. 1997. Network Fragments: Representing Knowledge for Constructing Probabilistic Models. In *Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference*, 334-341. San Mateo, CA: Morgan Kaufmann Publishers.

Sato, T. 1998, Modeling scientific theories as PRISM programs. In *ECAI-98 Workshop on Machine Discovery*, 37-45. Hoboken, NJ: John Wiley & Sons.

Wellman, M., Breese, J., Goldman, R. 1992. From Knowledge Bases to Decision Models. *Knowledge Engineering Review* **7**:35–53.