Truth Maintenance System with Probabilistic Constraints for Enhanced Level Two Fusion

Subrata Das and **David Lawless**
Charles River Analytics, Inc.
625 Mt. Auburn St.
Cambridge, MA 02138, USA
+1 617 491 3474
{sdas,dlawless}@cra.com

Abstract - *A Network-based Truth Maintenance System (NTMS) is presented that leverages Bayesian Belief Network (BN) technology for its truth maintenance mechanism. NTMS applies probabilistic versions of integrity constraints in the form of small BNs to the knowledge base, providing an efficient and intuitive mechanism for detecting inconsistencies. The approach is applicable to any relational database by converting database entries into evidence for the probabilistic constraints. NTMS can optionally integrate BNs for Level Two fusion (Situation Assessment, or SA) with the probabilistic constraints; the instantiation for a particular problem consists of a SA BN that models the world state and facilitates SA on the knowledge base. The SA BN is augmented with the probabilistic constraint BNs, which provide metrics on the consistency of the SA BN. Inconsistencies in the SA BN are resolved by a combination of techniques including sensitivity analysis, default reasoning, consistency 'forcing', or seeking new evidence.*

Keywords: *Truth Maintenance System, Bayesian Belief Networks, probabilistic constraints, default reasoning*

1 Introduction

Contemporary digital information systems aggregate enormous amounts of data for use in critical situation analysis and decision making tasks; however, the overall accuracy and consistency of the databases can be very difficult to verify and maintain. For example, military information systems may concentrate millions of reports from vast and disparate sensor arrays (including, to name just a few, unattended ground vibration sensors, thermal sensors, radar, UAV video feeds, satellite data, human intelligence reports, etc.) for situation analysis purposes, such as determining enemy actions or intent; but due to the context sensitive, inexact, and uncertain nature of the incoming raw information, ensuring consistency of such a military intelligence database is especially difficult. The traditional AI solution to this problem is to employ a Truth Maintenance System (TMS) [5][7][15][16]; such systems are expressly designed to help maintain consistency in the knowledge base used by a problem solver. However, the overhead of TMS usage is usually prohibitive for large problems, and their truth maintenance is intractable in the general case.

This paper presents a Network-based Truth Maintenance System, or NTMS [3], a TMS variant that incorporates a *purely probabilistic approach* to database consistency, rather than a logic based approach. NTMS uses Bayesian Belief Networks (BNs) [20] as its fundamental enabling technology. NTMS introduces the use of *probabilistic integrity constraints* (or more simply, probabilistic constraints) with BNs for detecting inconsistencies; these can be combined with more standard mechanisms such as sensitivity analysis [12] and default reasoning [23] for consistency recovery purposes. The NTMS variant provides a practical methodology that facilitates a natural approach to maintaining database consistency, and avoids some of the overhead associated with other TMS systems [4].

1.1 Motivating Example

A simple example will illustrate the basic ideas behind the NTMS system. A knowledge elicitation (KE) session with a military subject matter expert (SME) regarding the analysis of reconnaissance reports produced the rule of thumb that "Enemy tanks are rarely found in the woods" (the rationale being that enemy tanks generally operate in open areas for better mobility, and usually only move into a wooded area for camouflage). The SME was further able to partially quantify his rule of thumb, providing estimates (see Table 1) of the probabilities of finding enemy equipment in a wooded area or on a road.

Table 1: Tank/Woods Consistency

Equipment	Terrain	p(Consistent)	p(Inconsistent)
Tank	Non-Woods	0.8	0.2
Tank	Woods	0.1	0.9
Non-tank	Non-Woods	0.5	0.5
Non-tank	Woods	0.4	0.6

Although SMEs often have little if any background in probability theory (let alone Bayesian Networks), we have found that they can usually provide general rules about the likelihood of world state combinations, which translate fairly easily into rough probability estimates,

which in turn are the basis of the probabilistic constraints used by NTMS. The above rule yields a probabilistic constraint assigning a probability of p(Inconsistent)=0.9 to any incoming reconnaissance report (or combination of such reports) indicating a sighting of an enemy tank in a wooded area. Depending on the system parameters in effect, such a metric of inconsistency would trigger recovery action by the NTMS system; recovery might then lead to rejection of the report or a prior report, to the revision of default assumptions about the terrain or equipment involved, or to a request for additional resolving information.

1.2 NTMS Overview

The block diagram in Figure 1 shows how the NTMS approach has been implemented in a prototype.

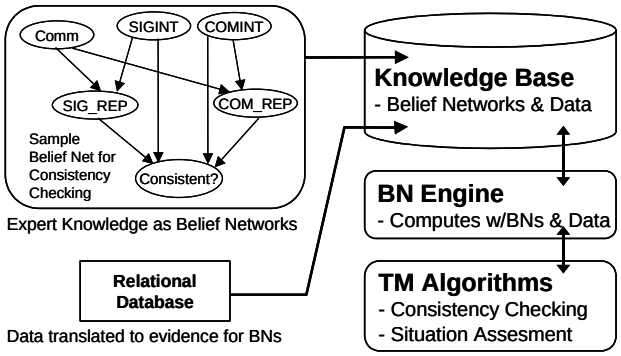

Figure 1: NTMS Block Diagram

As shown in the diagram, NTMS operates on a conceptual knowledge base of belief networks (BNs) and data. The BNs consist of Situation Assessment (SA) networks and probabilistic constraint networks; the SA networks are augmented (before run time) with small networks representing probabilistic constraints for consistency checking. The data consists of relational database entries which are translated (at run time, with assistance from additional small BNs for low-level, i.e. Level 1, data fusion[1]) into BN evidence; the evidence is then posted to the augmented SA networks to update the internal world state model. The BN engine is a software component that handles BN inferencing and probability calculations for the augmented SA networks. The truth maintenance algorithms use the BN engine to aggregate evidence, and perform situation assessment, consistency checking, and recovery (when inconsistency exceeds some specified threshold).

1.3 Related Work

In logic or deductive databases without probability, missing values are represented by Skolem constants; more generally, in logic programming missing values, or existentially-quantified variables, are represented by terms built from Skolem functors. There has been similar research in embedding probabilities into both conventional relational databases [8][10] and deductive databases [18]. In addition, embedding probabilities into logic programming [14][19][21] is also relevant to our research, but not the more general approach of embedding probabilities into logic [6]. Each of these systems has query processing capabilities, and thus integrity constraints can be defined and verified upon update via transaction.

For example, Fuhr and Rolleke [8] propose a probabilistic relational algebra (PRA) which is a generalization of standard relational algebra. In PRA, tuples are assigned probabilistic weights giving the probability that a tuple belongs to a relation. A probabilistic relational model [10] uses a Bayesian network to represent the joint probability distribution over fields in a relational database. Like our approach, the network can be used to make inferences about missing values in the database. Ng [18] proposes a deductive database framework which is expressive enough to represent such probabilistic relationships as conditional probabilities, Bayesian updates, and probability propagation. Probabilities in probabilistic logic programming by Lukasiewicz [14] are defined over a set of possible worlds and linear optimization techniques are employed for deciding satisfiability. Finally, Bokor and Ferguson [1] use an approach similar to ours to reason about diagrams.

Our approach differs from each of the above in terms of practicality and efficiency. Moreover, unlike conventional relational databases and most of the other approaches, we do not throw away a transaction due to the violation of an integrity constraint. We have also developed various ways of recovering from constraint violations.

Various types of truth maintenance systems (TMS) have been devised in the past, with different features according to their research aims [7]:

- A Justification-Based Truth Maintenance System (JTMS) enables one to examine the consequences of the current set of assumptions.
- An Assumption-Based Truth Maintenance System (ATMS) [13] allows one to maintain and reason with a number of simultaneous, possibly incompatible, current sets of assumptions.
- A Logic-Based Truth Maintenance System (LTMS), like a JTMS, reasons with only one set of current assumptions at a time, but recognizes the propositional semantics of sentences, that is, understands the relations between p and not p, p and q and p or q, and modal operators for knowledge and belief [9].

The NTMS variant is similar to a JTMS because we examine the consequences of one piece of evidence at a time. It can be easily extended to an ATMS by replicating

[1] The terms *data fusion*, *situation assessment*, and the various fusion levels follow the JDL definitions [11].

the SA network to simultaneously handle a set of incompatible assumptions. However, we have avoided the LTMS approach, as any constraint satisfaction procedure required to implement such a system is NP hard and therefore unsuitable for handling the large information sources we are targeting (i.e. military intelligence databases). Table 2 compares the NTMS approach with that of the logic-based Expert Systems mentioned above, and also with a traditional Database Management System (DBMS), considered as a TMS with its deterministic integrity constraints.

Table 2: TMS Feature Comparison

TMS Type / TMS Feature	Database Management Systems (DBMS)	Logic-based Expert Systems (ES)	Belief Network Engine Systems (NTMS)
Knowledge Base	Relational facts and views	Horn/general clauses	Belief networks and evidence
Inference Engine	Query answering	Theorem prover	Probabilistic Reasoning via Evidence propagation
Truth Maintenance Algorithm	Integrity Constraint verification	JTMS, ATMS, LTMS	Probabilistic Constraints
Inconsistency Detection & Recovery	Violation of constraints and discard update	Derivation of 'nogood' and revised earlier assumption	Network sensitivity analysis
Default Assumptions	Closed World Assumption	Default assumption	A priori beliefs
Backtracking	None	Dependency driven	Node links and strength

2 Probabilistic Constraints

2.1 Definition

Integrity constraints are a standard mechanism for ensuring the integrity of databases. A discussion on various types of constraints and verification methods for relational and deductive databases can be found in [2][17]. Such a database must 'satisfy' its integrity constraints. In the theoremhood view of satisfaction, a database is considered as a logical theory, and each constraint must be a theorem of the theory. In the consistency view, the constraints along with the theory must be logically consistent. We extend this notion of integrity constraints to probabilistic integrity constraints (just probabilistic constraints or even constraints when the context is clear). However, there are important differences between the deterministic and probabilistic varieties.

A standard deterministic integrity constraint is boolean, in the sense that it assigns a world state to a range of just two values, i.e. it is either completely consistent (true) or completely inconsistent (false). Probabilistic constraints assign a world state to the unit interval, so that a world state may be completely consistent (1.0), completely inconsistent (0.0), or anywhere in between these extremes.

The unit interval value assigned to a world state is expected to estimate the actual probability of occurrence of the world state, as a metric of 'consistency', e.g. p=0 would represent an impossible world state (a contradiction or complete inconsistency), and p=1 would represent a certain world state (a tautology, completely consistent).

Deterministic constraints on world states are satisfied when they evaluate to true. Probabilistic constraints are satisfied whenever their computed value exceeds the SME-specified threshold; they are therefore highly subjective.

Finally, probabilistic constraints use Bayesian Belief Networks (BNs) [20] as their specific form of knowledge representation. (This is essentially an implementation dependent choice; equivalent representations may serve better in other implementations, e.g. a set of rules might be preferable for an expert system.)

2.2 Ontology

A prerequisite task in formulating constraints for a particular problem (or scenario) is listing the entities (and their properties or attributes) that are to be constrained, that is, one needs to specify the *ontology* of the scenario, so that a fixed and well-defined set of discussion terms, and hence scenario world state variables, exists.

Although not especially difficult, the task of specifying an ontology for a scenario can be laborious and time-consuming. Moreover, the ontology needs to be specified in a way that makes it available to all the software components of the NTMS system. For the prototype implementation, we used the freely available Protégé ontology editor [22] for this purpose.

2.3 Representation

The formation of probabilistic constraints basically consists of knowledge elicitation (KE) with subject matter experts (SMEs) to obtain probability estimates for world state combinations, followed by translation to a BN knowledge representation (KR). So continuing our example from above, Table 1 specified the requisite probabilities for a probabilistic constraint on enemy tanks in the woods; we can readily translate the information to a BN, forming the Tank/Woods probabilistic constraint of Figure 2. Note that the table translates directly into the Conditional Probability Table (CPT) of the *Consistent* node of the BN representation.

Probabilistic constraints may involve any number of variables; Figure 3 shows an example constraint on three variables, which simply checks that a military unit (e.g. a squad or platoon) comprised of a certain equipment type (e.g. tanks or artillery) has a size that is consistent with its unit designation. In general, probabilistic constraints on large numbers of variables should be avoided because a) the KE process becomes much more difficult for such cases, and b) the CPT size is exponential in terms of the

number of variables involved. Fortunately, in practice only two- or three-variable constraints usually arise; probabilistic constraints on more variables are rare, and moreover can often be effectively reduced to a collection of constraints on fewer variables via *ad hoc* techniques.

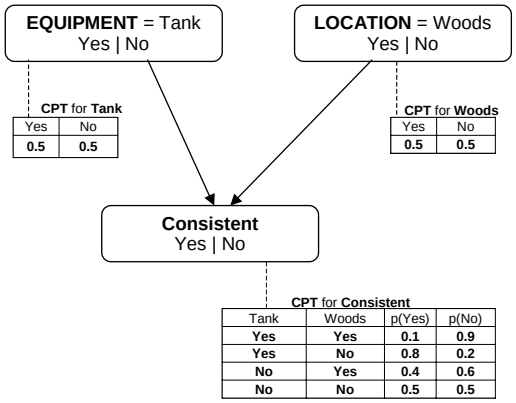

Figure 2: Tank/Woods Probability Constraint

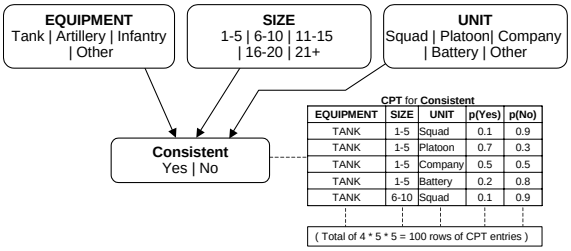

Figure 3: Probabilistic Constraint on Three Variables

2.4 Hierarchy

The various probabilistic constraints that arise will often be minor variations from one another; this similarity can be leveraged in the KR format by applying standard object-oriented programming techniques. We treat each constraint as an object or class, and organize a class hierarchy of constraints. The higher level constraint objects will represent more abstract constraints; with more specific constraints represented in subclasses.

For example, a military SME may wish to represent the simple, well-known constraint that barbed wire and minefields are usually collocated in older, "conventional" battlefield situations. Figure 4 shows how this constraint might fit into a constraint class hierarchy.

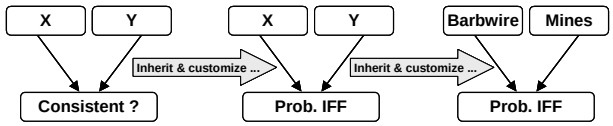

Figure 4: Class Hierarchy of Probabilistic Constraints

The highest level constraint, on the left, is simply an abstract representation of two entities X and Y, with an unspecified consistency relationship between them. The next level, in the middle, refines the abstract constraint somewhat to a probabilistic if-and-only-if (IFF) constraint on two variables, indicating that X and Y should usually occur together. Its CPT will indicate a high degree of consistency if X and Y both occur together or are both absent, and a low degree of consistency if exactly one of X and Y are present. Finally, the right-most constraint is specifically tailored for the case of barbed wire and mines being collocated, created by setting X=Barbwire and Y=Mines; the SME may also make adjustments to the inherited Conditional Probability Table (CPT).

3 NTMS Prototype

3.1 Algorithmics

The most salient features of the NTMS prototype algorithms include:

- SME-guided configuration and application of probabilistic constraints to a Situation Assessment Belief Network (SA BN) for truth maintenance (TM).
- Integration into a single BN of Level 1 fusion (state estimation) BNs, Level 2 fusion BNs (SA BN), and probabilistic constraint BNs.
- Application of the augmented SA BN to incoming sensor data for consistency checking and error detection.
- Techniques for recovery from constraint violations.

The configuration of probabilistic constraints is an important KE task (discussed at length in Section 3.2); their application to the SA BN is accomplished by a straightforward selection of applicable constraints for a given SA BN from a library, guided by a program menu.

The integration of Level 1 BNs, the SA BNs, and the constraint BNs is illustrated in Figure 5. In this example, the SME manually adds eight nodes: four for the SA task he wishes to perform, and another four for Level 1 fusion input that he requires for the SA task. At run time, fifteen additional nodes are added to the network to perform Level 1 fusion; in this case the nodes are for discretization of input from the auxiliary GIS subsystem. (Nodes handling Level 1 fusion have special logic associated with them for this task.) Additionally, three probabilistic constraints selected by the SME are appended to the initial SA BN.

The augmented SA BN is then applied to incoming sensor reports by using the Level 1 fusion nodes to discretize the information required from the sensor reports, which is effectively posted as evidence to the nodes. (Similarly, for the GIS-related nodes, location information is extracted from the sensor reports, and the GIS subsystem is consulted for information that is discretized by the nodes and posted as evidence.)

The posting of evidence to the augmented SA BN propagates through all nodes, and so affects the belief states of the constraint nodes. If some constraint falls below its threshold consistency, an alarm condition is signaled, and recovery techniques are invoked; these

techniques are important algorithms that are discussed at length in Section 4.

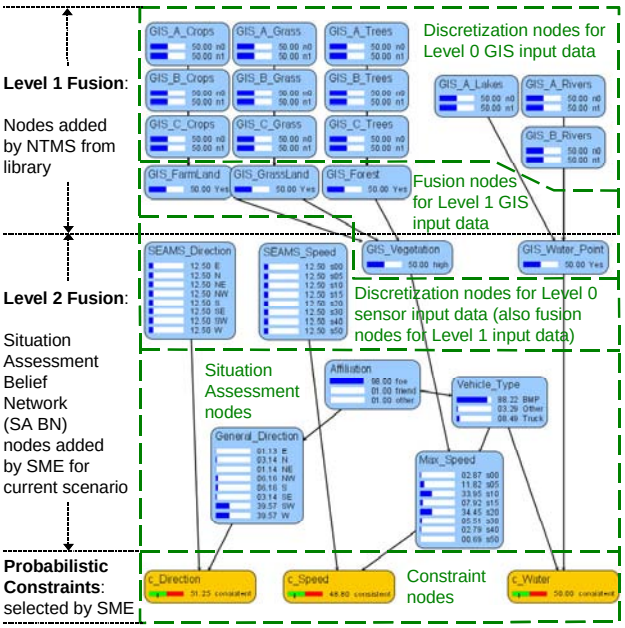

Figure 5: Fully Augmented SA BN in NTMS

3.2 KE for Constraints

One goal of the NTMS research was to provide easy knowledge elicitation (KE) of probabilistic constraints; we hoped to make the KE process accessible to the subject matter experts (SMEs) we consulted, so that they could personally review and revise the captured constraint. To illustrate the KE process devised, we will again use our more-or-less standard constraint example that tanks are generally only found in certain terrains: they are usually found on plains or other flat wide spaces, only occasionally found in wooded areas due to the difficulty of maneuvering there, and almost never found in the water (for a functional tank this may possibly occur if marine transport is in use).

To formalize the constraint, i.e. prepare it for use with the computer, the SME decides what discrete variables to use, and what states they should have. That is, the SME first determines the ontology terms that will be used. For this constraint, we use the variable *Terrain* with states *water*, *plains*, *woods*, and the variable *Tanks* with states *few* and *many*. To capture the SME's knowledge, we need to have him specify the degree of consistency of all possible combinations of the two variable's states. Terrain has 3 states, and Tanks has 2 states, so there are 6 possible combinations to be quantified; these 6 state combinations are shown in the 2 leftmost columns of the constraint configuration interface in Figure 6.

The next step in capturing the constraint is to have the SME decide how consistent or inconsistent each state combination is. With the interface we provided, the SME can adjust the consistency for each combination by simply moving the slider for that combination to the desired level of consistency, which can be anything from absolutely consistent to absolutely inconsistent, as shown in Figure 6. We feel this is an easy and intuitive way of specifying probabilistic constraints (although the work can become tedious when there are many states to quantify). In the figure it can easily be seen by checking the slider bars that the SME has specified that whenever tanks are found in the water it is essentially inconsistent, when tanks are found on the plains this is essentially a consistent situation, and when tanks are found in the woods the consistency level is something in between absolutely consistent and absolutely inconsistent, although sighting many tanks in the woods is less likely to be consistent than sighting just a few.

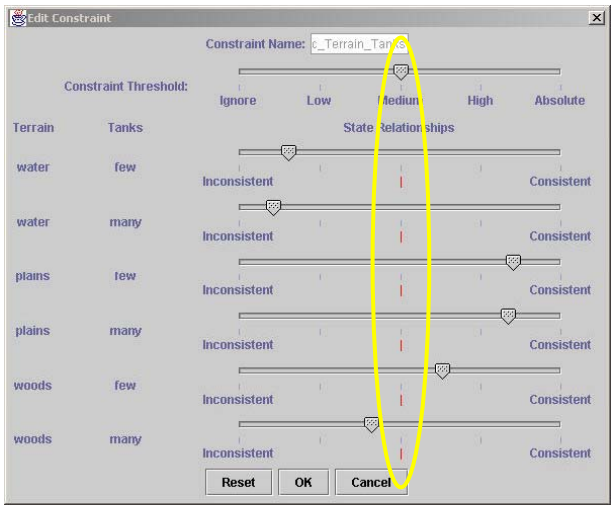

Figure 6: Constraint Configuration Interface

After the consistency levels have been specified for each state combination, the constraint is complete, and can be saved for future use in any number of scenarios or assessment situations. For application in a particular scenario, one final step is needed, which is to specify just how stringently the constraint should be applied to the scenario. This parameter is needed because, in practice, minor inconsistencies will very often occur in incoming field reports, and it is generally more useful to simply permit such inconsistencies rather than signal a serious problem. For our sample constraint, a report of a few tanks in the woods may arrive, which per the SME is not especially consistent (and hence may warrant further investigation), but this issue likely has lower priority than the overall situation assessment task under way, such as where the enemy force is headed, and so some allowance for this must be provided. This is done by providing a specific threshold for the required degree of consistency of a constraint: if the consistency level as determined by the constraint is above the threshold the truth level is acceptable so nothing is done, but if it falls below the threshold due to incoming evidence (field reports), a warning is signaled. In Figure 6 the large yellow oval shows that a medium level threshold has been specified, which means that as long as the calculated consistency is

above this level, no warning will be raised. An intuitive feature of the interface is that it shows at a glance which state combinations will cause the consistency warning to occur: those to the left of the vertical red line (inside the large yellow oval) marking the threshold level will result in a warning, while those to the right of the line will not. Hence a few tanks sighted in the woods would not cause a warning, however many tanks sighted in the woods would result in such a warning indication. Also, any report of tanks in the water would also raise a warning indication.

The constraint threshold should be selected carefully for a given scenario; the threshold level specified would depend on how important it is for the constraint to be satisfied. If the threshold level is set too high (near 'Absolute'), then the inconsistency warning signal would nearly always be given, since no matter what state combination is effective, the resulting consistency will nearly always be below the specified threshold. It is also possible to apply a constraint too liberally: if the threshold is set too low (near 'Ignore'), the constraint would nearly always be satisfied, and so would serve little or no useful purpose in truth maintenance.

3.3 Context Resolution

An important issue in applying probabilistic constraints with the NTMS approach is that of ensuring proper context for the constraints. When constraining the combinations of two or more variables, the variable data must occur in the same context. For example, using our earlier Tank/Woods constraint, one incoming record (from the military database or other source) may indicate a tank sighting, and another may indicate a wooded area. Application of the Tank/Woods constraint to the combined evidence from these two records would only be valid if the records referred to the same event, i.e. the records must have compatible context.

Out-of-context records will often occur due to temporal differences, where different records occur in the context of different times. Similarly, records referring to different geographical areas often cause a context mismatch. Generally speaking, a context mismatch may occur any time two records have an implicit (rather than explicit) attribute whose values differ.

The NTMS research effort did not attempt to solve the problem of context resolution, which is an open area of research. A circumvention employed in the prototype was to restrict constraints to evaluating consistency on a per-record basis, which virtually guarantees proper context for evaluation.

3.4 Current Implementation, Future Work

The block diagram of Figure 7 illustrates the software architecture of the NTMS prototype. Incoming sensor messages (which are replayed from log files in the case of simulations) are concentrated in a military data hub known as the Sensor Exploitation And Management System (SEAMS). The SEAMS hub uses an SQL database for backing storage, and communicates with subscribing client applications such as NTMS via a CORBA server. The Java-based NTMS application adds location-dependent information via a custom GIS subsystem, and performs situation assessment and truth maintenance (including constraint checking) with the assistance of a tightly coupled belief network engine.

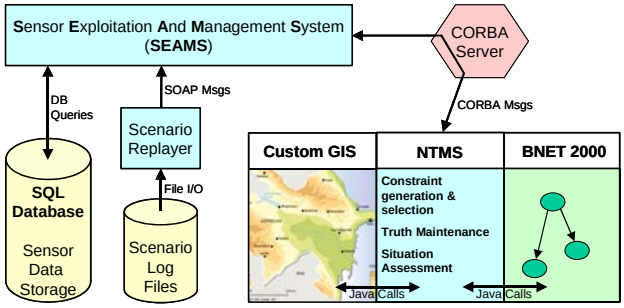

Figure 7: NTMS Software Architecture

The prototype throughput was approximately 10 sensor records per second. This included notifying the SEAMS server about occasional inconsistent reports. For operational use, it is estimated that a throughput of about 100 records/second would be needed. We project that the gap between the prototype and operational performance can be closed by reducing network delays, using faster computers, re-implementing in C++ (instead of Java), and general re-factoring of the code base.

Aside from improving performance, future plans include addressing the context resolution issue discussed above (although this is more of a research issue than a development issue). Additionally, we feel that probabilistic reasoning by itself is clearly not sufficient to solve all the problems of TM, hence we plan to investigate some practical form of integration between probabilistic and logic-based systems. Finally, we plan to move towards an operational system, by conducting KE to assemble a complete and easily accessible library of probabilistic constraints and situation assessment networks.

4 Recovery from Inconsistency

4.1 BN Sensitivity Analysis

Sensitivity analysis [12] (also referred to as mutual information in some texts) is a computational procedure on BNs that is leveraged as a fundamental tool by NTMS. NTMS uses sensitivity analysis as a search aid, to help isolate groups of nodes that may be causing inconsistency in the SA BN due to errors in their posted evidence. Given a particular node X in the fully augmented BN, sensitivity analysis determines which nodes Y_i in the BN most affect X; i.e. it finds and ranks the Y_i which cause the greatest change in X's belief values as evidence is changed in a particular Y. In a large, fully augmented SA BN, the Y_i may be far (in terms of links) from X; moreover, the procedure is highly dependent on the

current BN state, in that different posted evidence leads to different nodes (and rankings) Y_i.

Given a node X whose belief state is suspect - usually an NTMS probabilistic constraint node - and the node Y found by sensitivity analysis that most affects X; we use a simple heuristic approach that Y is therefore the *most likely cause* of the error at X. This approach can be extended to isolate entire groups of nodes which may be causing inconsistency or error, e.g. by selecting the top N ranked nodes Y_i to which X is sensitive.

4.2 Default Reasoning

Default reasoning [23] is a truth maintenance technique from formal logic, adapted for use with BNs and probabilistic reasoning. It involves considering and revising the space of all possible default settings to achieve a particular desired result involving the network's belief values. For BNs the default settings are the *a priori* probabilities set in the root nodes of the networks; these settings represent reasonable assumptions for the current scenario, made by the SME. There may be many such nodes in a large, fully augmented SA BN used by NTMS; sensitivity analysis can be used to narrow the choices to a workable set.

NTMS use of default reasoning yields a general TM algorithm for restoring consistency, which we sketch here:

1. When a constraint node X falls below its specified threshold, run sensitivity analysis on X to identify and rank those BN nodes Y_i which most affect X.
2. From the list of nodes Y_i select those nodes Z_i which are root nodes, as these have the default assumptions, encoded as *a priori* probabilities.
3. Form the state space S consisting of all possible likelihood (evidence) settings for the Z_i.
4. Search S for a combination of likelihoods on the Z_i that will raise the consistency of X back above its specified threshold. (We may wish to find a minimal subset of the Z_i to alter to restore consistency.)
5. Post the restorative likelihoods and continue.

Note this algorithm is not guaranteed to succeed: we may fail to find any root nodes on our list of nodes affecting the violated constraint, or it may be that no combination of likelihoods from the state space S will restore consistency.

Figure 8 illustrates default reasoning.

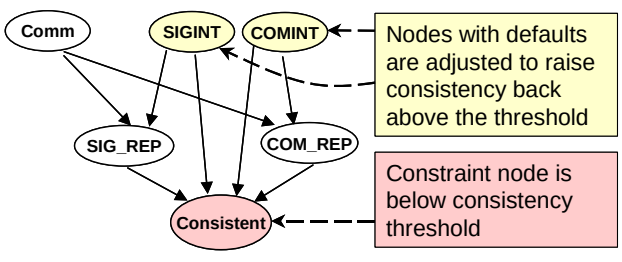

Figure 8: Default Reasoning Example

Given a constraint violation at X='Consistent', we perform sensitivity analysis to find the list Y_i of nodes affecting X, then select from the Y_i the root nodes Z_1='SIGINT' and Z_2='COMINT'. We then try to restore consistency in the BN by adjusting the defaults (via posting evidence/likelihoods) to 'SIGINT' and 'COMINT'.

4.3 Consistency Forcing

If default reasoning fails to restore consistency, we may try an approach we call consistency forcing; this works as follows (as above, this is just a sketch of the algorithm):

1. When a constraint node X falls below its specified threshold, simply post evidence to X sufficient to 'force' its consistency above the threshold.
2. Examine all other non-constraint nodes in the network: compute the Euclidean distance between their belief state vectors before and after forcing consistency on X, and rank them in descending order, in a list Y_i of such nodes. (In many cases, we will want to examine only those nodes with evidence posted.)
3. Retract the evidence previously posted to X.
4. In order (i=1,…,n), post evidence to the Y_i to bring their belief state vectors 'close' to what they were with evidence posted to the constraint node X. Continue until the consistency of X is back above threshold, or until we exhaust the Y_i.

Unfortunately, this algorithm is also not guaranteed to succeed: in some unusual limiting circumstances, certain combinations of evidence are fundamentally incompatible, so that the BN engine will reject their application.

Consistency forcing can be very effective as a TM algorithm, as it effectively widens the field of nodes whose beliefs we can manipulate to essentially the entire BN, save for the constraint nodes. On the other hand, there may in many cases be no acceptable semantic interpretation of modifying the belief states of the non-default (i.e. non-root) or non-evidentiary (those with no evidence posted) nodes; whereas with default reasoning it is nearly always acceptable to interpret the changes as simply 'correcting bad assumptions.' For this reason it is usually better to try default reasoning first.

4.4 Last Resorts

If neither default reasoning nor consistency forcing (or any variation thereof) restore consistency to the SA BN, then there are two remaining possibilities:

1. The incoming evidence is fundamentally inconsistent.
2. The SA BN model is inaccurate.

Distinguishing between these two cases in the field can be difficult; we recommend avoiding the latter by carefully modeling, and then carefully verifying the BN model before fielding it.

Given the former case, i.e. we have been provided with inconsistent evidence, some coping tactics include:

1. Simply reject the inconsistent record outright. The NTMS prototype will raise an error dialog to warn of (rather than reject) incompatible evidence records.
2. Revise the probabilistic constraint threshold levels. In some cases it may be preferable to simply tolerate some inconsistency.
3. If multiple input records have been posted to the SA BN, it may be possible to identify a single record (not necessarily the most recent) that is incompatible with the rest, and whose veracity is suspect: this record can be rejected and the others retained.
4. Wait for more evidence. In many applications incomplete sets of information will seem inconsistent; all records must be received to restore consistency.
5. Ask for assistance. A human-in-the-loop hybrid TM approach could be devised which asks the SME to assist in restoring consistency by choosing which subsets of consistent evidence are more likely correct.

5 Conclusion

The NTMS system attempts to incorporate simple, human-like probabilistic reasoning into a TMS in an efficient, practical, and accessible way. NTMS relies entirely on probabilistic reasoning for its truth maintenance, which breaks with the traditional coarse grained handling of consistency enforcement via logical formulae.

NTMS uses BN models, performing probabilistic situation analysis (SA) as specified by the SME who creates the BN. It extends the probabilistic reasoning paradigm by using probabilistic constraints to check the consistency of incoming evidence, and a probabilistic variant of default reasoning and other techniques to restore consistency in the event of constraint violations.

NTMS does not address issues of context resolution; proper context must be provided by the system user to ensure proper results. The NTMS prototype restricts its reasoning to the verification of individual records to ensure proper context of the data elements.

NTMS has been implemented on a military command and control (C2) platform, and tested with computer simulations. Future work includes addressing performance issues, the context resolution problem, some integration with logic-based systems, and expanding the knowledge base of constraints and assessment networks.

Acknowledgment: The authors wish to thank Mr. Joseph A. Karakowski, of the US Army CECOM, at Fort Monmouth, NJ, for his support on this project.

References

[1] Bokor, J., and Ferguson, R., *Integrating Probabilistic Networks into a Symbolic Diagrammatic Reasoner*, 18th International Workshop on Qualitative Reasoning, Northwestern University, August 2004.

[2] S. K. Das, Deductive Databases and Logic Programming, Addison-Wesley, 1992.

[3] S. K. Das, and Lawless, D. *Network-based Truth Maintenance System*, Proceedings of the 15th European Conference on Artificial Intelligence, pp. 551-555, 2002.

[4] Dechter, R., and Dechter, A., *Structure-Driven Algorithms for Truth Maintenance*, Artificial Intelligence, Vol. 82, No. 1-2, pp. 1-20, 1996.

[5] J. Doyle, *Truth Maintenance Systems*, Artificial Intelligence, Vol. 12, No. 3, pp. 231-272, 1979.

[6] R. Fagin, J. Y. Halpern, and N. Megiddo, *A Logic for Reasoning about Probabilities*, Information and Computation, Vol. 87, No. 1&2, pp. 78-128, 1990.

[7] K. D. Forbus, and J. de Kleer, *Building Problem Solvers*, MIT Press, 1993.

[8] N. Fuhr, and T. Rolleke, *A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems*, ACM Transactions on Information Systems, Vol. 15, No. 1, pp. 32–66, 1997.

[9] P. Gardenfors, Knowledge in Flux: Modeling the Dynamics and Epistemic States, MIT Press, 1988.

[10] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer, *Learning Probabilistic Relational Models*, in Relational Data Mining, Springer-Verlag, 2001.

[11] D. L. Hall, and J. Llinas, *Handbook of Multisensor Data Fusion*, CRC Press, 2001.

[12] F. V. Jensen, *An Introduction to Bayesian Networks*, Springer-Verlag, 1998.

[13] J. de Kleer, *An Assumption-based TMS*, Artificial Intelligence, Vol. 28, No. 2, pp. 127-162, 1986.

[14] T. Lukasiewicz, *Probabilistic Logic Programming with Conditional Constraints*, ACM Transactions on Computational Logic, Vol. 2, No. 3, pp. 289-339, 2001.

[15] D. McAllester, *Truth Maintenance*, Proceedings of the Eighth National Conference on Artificial Intelligence, Vol. 2, pp. 1109-1116, 1990.

[16] D. McDermott, *A General Framework for Reason Maintenance*, Artificial Intelligence, Vol. 50, No. 3, pp. 289-329, 1991.

[17] J. Minker, *Logic and Databases: Past, Present, and Future*, AI Magazine, Fall 1997.

[18] R. Ng, *Reasoning with Uncertainty in Deductive Databases and Logic Programs*, International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, Vol. 2&3, pp. 261-316, 1997.

[19] R. Ng, and V. S. Subrahmanian, *Probabilistic Logic Programming*, Information and Computation, Vol. 101, No. 2, pp. 150-201, 1992.

[20] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.

[21] D. Poole, *Probabilistic Horn Abduction and Bayesian Networks*, Artificial Intelligence, Vol. 64, No. 1, pp. 81-129, 1993.

[22] Protégé Ontology Editor, http://protege.stanford.edu/

[23] R. Reiter, *A Logic for Default Reasoning*, Artificial Intelligence, Vol. 13, pp. 81-132, 1980.