135323

p- 19

A Beginner's Guide to Belief Revision and Truth Maintenance Systems

Cindy L. Mason
Artificial Intelligence Research Branch
MS 269-2
NASA Ames Research Center
Moffett Field, CA 94035-1000

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA-92-33

October, 1992

A Beginner's Guide to Belief Revision and Truth Maintenance Systems

Cindy L. Mason

1 Introduction

Over the past decade, belief revision systems and truth maintenance mechanisms have proven to be of general utility in a wide variety of problems. Although truth maintenance is widely discussed in the literature, most authors assume familiarity with the ideas of truth maintenance systems and belief revision. This document serves as a primer on the basic notions of truth maintenance and belief revision. It intentionally avoids use of the often confusing terminology found in the TMS literature and concentrates on the ideas of TMSes. We first describe the general notion of reasoning with assumptions, and the idea of beliefs. Next, we describe the traditional programming model for systems that deal with beliefs. Subsequently we explore the JTMS and ATMS in detail. A bibliography of TMS literature can be found in [Martins 1990].

Assumption-based (or default) reasoning is concerned with inference from assumptions (defaults). This type of inference has been identified with the following patterns of human reasoning [Reiter 1988]:

Normally, P holds.

Typically, P is the case.

Assume P by default.

where conclusions are based on knowledge of what is typical, or usual, when there is not information to conclud therwise. For example,

Normally, Men like Women.

Typically, Men like Women is the case.

Assume Men like Women by default.

This so called *plausible* reasoning generates new beliefs from old ones using assumptions, and is not purely deductive in the classic sense. Assumptions and consequent conclusions are tentative, since subsequent inferences and information can lead to contradictions involving the assumptions.

The canonical example of this type of reasoning (modernized in [Ginsberg 1987]) is the following: If we know that most birds fly, and we know Opus is a bird, then we may reasonably conclude that Opus can fly. Furthermore, we can conclude that when Bill shoves Opus over the edge of the Grand Canyon, Opus lives, because he can fly.

The general idea is that we can use an assumption (or default conclusion) and the inferences it entails as long as there is no information to indicate these beliefs are wrong,[1] as formulated in [Reiter 1980], "in the absence of any information to the contrary, assume ..." A reasoning agent may use its assumptions and consequent deductions as long as they are consistent with what is known[2].

The difficulty in implementing assumption-based reasoning is that default assumptions and their consequents are tentative. Facts that are added might later be thrown out. In our example, if we find out later that Opus is a penguin, the original assumption and subsequent inferences must be revised. The assumption that Opus can fly must be deleted because penguins can't fly, and the conclusion that Opus lives after he was shoved into the canyon must be deleted as well, since it depends on the assumption that Opus could fly.

The term *belief* is often used in the context of default reasoning to denote the idea that the conclusions of the reasoning agent are subject to revision, hence the term *belief revision system.* This type of reasoning is *non-monotonic* since some of the beliefs may be withdrawn as reasoning pro-

[1]This type of reasoning can be traced back to the THNOT operator in MICRO-PLANNER[Sussman et. al. 1970].

[2]By 'known' we refer to the facts presently in the fact base, rather than to the set of facts which may be derivable. In general, determining whether or not a fact can be derived is not decidable.

gresses and the totality of beliefs need not increase monotonically.

A truth maintenance system (TMS)[3] is the collection of procedures and data structures used for accomplishing belief revision. Rather than actually removing fact base items, the TMS marks each fact base item to indicate whether or not it is currently *BELIEVED*. The task of the truth maintenance system is to maintain the set of conclusions that are believed at any point during reasoning.

TMS's are recognized as the most important component of systems that can reason with assumptions and revise beliefs over the course of problem solving. In general, belief revision techniques appear useful in any circumstance involving the need to maintain consistency or to consider multiple competing views. TMS's have been applied to a broad variety of areas, including software requirements verification[Chandra 1991], user modelling [Jones and Millington 1988], circuit analysis [deKleer and Williams 1987], vision [Provan 1987][Bowen and Mayhew 1989], signal interpretation [Mason et. al. 1988] [Mason and Johnson 1989], scientific discovery[Rose and Langley 1986], and analog circuit design [Stallman and Sussman 1977].

2 Architecture of the TMS

The TMS is commonly viewed as an independent program component associated with the automated reasoning system or *problem-solver* component.[4] The task of the problem-solver is to draw inferences based on its beliefs, while the job of the TMS is to maintain the consistency of the problem solver's beliefs.

Because this set of currently held beliefs largely controls the course of problem solving, the interface between the problem solver and TMS is an important aspect of a belief revision system. Most often the problem solver provides the TMS with input consisting of the results of each of its inferences - both the conclusions and the justifications. For example, "conclusion q is

[3]TMS is defined here as in [McAllester 1980] to mean a class of algorithms. The term was originally coined by Doyle in [Doyle 1978][Doyle 1979] to describe the first domain independent truth maintenance system, and was later renamed a Reason-Maintenance System (RMS) [Doyle 1983]

[4]Some belief revision systems incorporate truth maintenance functions directly into the overall problem-solver.

justified by belief in p_1, p_2, ... p_n " or "conclusion q is justified by belief in p_1, p_2, ... p_n and disbelief in r_1, r_2, ... r_n" to indicate conclusion q depends not only on beliefs but on disbelief.[5] The second justification is called a *non-monotonic justification* since the set of problem-solver beliefs may actually be reduced in size if the system should subsequently come to believe in any of r_1 ... r_n. The problem-solver component also informs the TMS when a contradiction has been encountered during inferencing.

The TMS, in turn, uses the input to record the beliefs (and disbeliefs) on which each conclusion depends. These explicit data dependency records are the principle data structures on which the process of belief revision is based. In general, belief revision occurs whenever the problem solver informs the TMS that conclusions are incompatible (i.e. a contradiction has occurred) or when assumptions or premises need to be withdrawn or changed. The problem solver may then query the TMS as to which propositions it can believe. From this perspective, the TMS can be viewed as a clause management system, providing yes/no answers to problem-solver queries of the form (believed? q).

The manner in which dependencies are recorded divides the TMS literature among two broad categories, the *justification-based* (or *single-context*), JTMS, and the *assumption-based* (or *multiple-context*), ATMS.[6] JTMS's record information about the facts that directly infer a fact. ATMS's record information about the default assumptions that produce a fact. This difference in recording style allows the ATMS to maintain multiple contexts of belief during reasoning while JTMS-type systems maintain only a single context. The ATMS achieves this by explicitly representing the sets of assumptions for which each fact is believed. Each type of TMS is discussed in the following subsections.

[5]The notion of disbelief may be implemented in many ways, including giving each item a belief status (IN or OUT, Believed or Disbelieved, etc.), absence of the belief in memory, or presence of the negated belief.

[6]A notable third category would be the LTMS, or Logic-based TMS [McAllester 1980]. A clear description of LTMS systems is presented in [Rich and Knight 1991].

3 Justification-based TMS

Doyle [Doyle 1979] built the first domain independent JTMS. All subsequent work in the field can be viewed as based on it.

Representing Beliefs and Reasons for Beliefs in a JTMS

The internal representation for a problem-solver fact is a TMS *node*. Each TMS node corresponds to a problem-solver fact. This abstraction allows the TMS to work on the dependency information without regard to a particular fact representation. A JTMS node has the form

<Node-Id, Fact, Justification, Belief-Status>

where Node-Id is the TMS node identifier used by the JTMS, Fact is the problem-solver fact to which this TMS node corresponds, and Belief-Status indicates whether the fact is BELIEVED or DISBELIEVED. Description of the Justification follows.

The justification for a fact, f, is represented internally as two lists of TMS nodes.

(<BELIEVED-LIST> <DISBELIEVED-LIST>)

A justification is *valid* when all nodes in the BELIEVED-LIST are BELIEVED and all nodes in the DISBELIEVED-LIST are DISBELIEVED.

Each TMS node has a belief-status represented by one of two labels:

- BELIEVED - if there is a valid justification for the fact.
- DISBELIEVED - if there is no valid justification for the fact.

At any point during problem solving, some set of facts in the fact base will be labeled BELIEVED and some will be labeled DISBELIEVED. The set of BELIEVED facts constitutes the current set of beliefs or current context. DISBELIEVED facts are not part of the currently held set of beliefs and are inaccessible to the problem-solver.

It is important to keep clear the distinction between the state of belief for a fact, as represented by the belief-statuses, and the interpretation, or truth value, of a fact. The TMS keeps track of derivations and does not determine

validity. The following observation should make this clear. Doyle's JTMS allows both A and NOT-A to each be represented as separate facts. Both may be BELIEVED, both may be DISBELIEVED, or one BELIEVED and one DISBELIEVED. It is up to the problem solver to notice when both A and NOT-A are BELIEVED and that there is an inconsistency.

A *premise* is a fact that is always believed, and is justified with an empty BELIEVED-LIST and empty DISBELIEVED-LIST. An *assumption* is a fact that requires the absence of other facts to be valid. Hence, any justification with a non-empty DISBELIEVED-LIST represents an assumption. This aspect of the JTMS directly implements the default-logic notion of tenative inference[Reiter 1980]:

$$\frac{BIRD(x) \quad \text{M:}FLY(x)}{FLY(x)}$$

where **M:** is the modal operator, "It is consistent to assume," and the form is read, "if x is a bird, and it is consistent to assume x flies, then infer x flies." All other justifications represent derived facts. The following example makes use of these concepts:

Node	Fact	Justification		Status
R1	If Bird(x) and M:Flies(x) Then Flies(x)	(() ())	Premise	B
R2	If Penguin(x) Then ¬Flies(x)	(() ())	Premise	B
R3	If Shoved_Into_Grand_Canyon(x) and Flies(x) Then Lives(x)	(() ())	Premise	B
P1	Bird(Opus)	(() ())	Premise	B
A1	Flies(Opus)	((P1,R1) (A2))	Assumption	B
A2	¬Flies(Opus)	(() (A1))	Assumption	D
P2	Shoved_Into_Grand_Canyon(Opus)	(() ())	Premise	B
D1	Lives(Opus)	((A1,P2,R3) ())	Derivation	B

Each line of the example has node identifier, the corresponding problem-solver fact it represents, the JTMS justification, and status of belief - B indicating BELIEVED, and D indicating DISBELIEVED. Rules R1-R3 are represented as premise nodes, as are facts P1 and P2. Nodes A1 and A2

correspond to the assumptions that were made in order to infer that Opus flies. Node D1 represents the derivation that Opus lives and is dependent upon assumption A1, premise P2, and rule R3. Assumption A1 depends on disbelief in assumption A2, and is valid as long as A2 has belief status DISBELIEVED.

At some later time, the observation that Opus is a penguin is entered into the system. This fact gives a valid justification to the assumption node A2, and causes node A1 to become DISBELIEVED. As a result, node D1 becomes DISBELIEVED as well. The resulting TMS nodes and their statuses are shown below:

Node	Fact	Justification		Status
R1	If Bird(x) M:Flies(x) Then Flies(x)	(() ())	Premise	B
R2	If Penguin(x) M:¬Flies(x) Then ¬Flies(x)	(() ())	Premise	B
R3	If Shoved_Into_Grand_Canyon(x) and Flies(x) Then Lives(x)	(() ())	Premise	B
P1	Bird(Opus)	(() ())	Premise	B
A1	Flies(Opus)	((P1,R1) (A2))	Assumption	D
A2	¬Flies(Opus)	(**(P3, R2)** (A1))	Assumption	B
P2	Shoved_Into_Grand_Canyon(Opus)	(() ())	Premise	B
D1	Lives(Opus)	((A1,P2,R3) ())	Derivation	D
P3	**Pengiun(Opus)**	(() ())	**Premise**	**B**

4 Assumption-based TMS

DeKleer observed that instead of associating an entire fact base with a single set of assumptions, one could associate sets of assumptions with each fact. This is the fundamental difference between a JTMS and an ATMS. For each problem-solver fact, an ATMS maintains an explicit list of assumption sets that support its derivation, and that characterize the reasoning contexts in which it is valid. In this respect, the ATMS is similar to Relevance Logic [Anderson and Belnap 1975], and to the TMSes of [Martins 1983] and [Williams 1984].

As a result of this view of contradictions and dependency information,

the problem-solver program can now consider multiple and possibly contradictory contexts simultaneously. Context switching is essentially cost-free, making the use of TMS mechanisms accessible to problem solvers that must explore a large percentage of potential solutions. In addition, problem solvers can reason about and compare multiple possible solutions at once. These aspects of the ATMS can be important in certain domains, such as multi-agent problem solving, where exploring the views of several agents is a requirement.

Representing Beliefs and Justifications for Belief

As with the JTMS, an ATMS "node" corresponds to a fact in the problem-solver's fact base. However, the information an ATMS stores with a node differs from that of a JTMS. While the JTMS associates a node with its justification, an ATMS associates each node with a label indicating the sets of assumptions (defaults) that support each valid derivation of the fact. The sets of assumptions are "the foundation to which every problem-solver datum can be ultimately traced"[deKleer 1986]. This idea can be illustrated by rewriting our JTMS example in terms of the ATMS:

Node	Fact	Label	
R1	If Bird(x) M:Flies(x) Then Flies(x)	{ {} }	Premise
R2	If Penguin(x) Then ¬Flies(x)	{ {} }	Premise
R3	If Shoved_Into_Grand_Canyon(x) and Flies(x) Then Lives(x)	{ {} }	Premise
P1	Bird(Opus)	{ {} }	Premise
A1	Flies(Opus)	{ {A1} }	Assumption
P2	Shoved_Into_Grand_Canyon(Opus)	{ {} }	Premise
D1	Lives(Opus)	{ {A1} }	Derivation

Here, each node is associated with the corresponding problem-solver fact, as before, but instead of an explicit label indicating belief status, each node is labelled with the sets of assumptions or reasoning contexts in which each fact was derived. Premises are valid in all reasoning contexts, and are labelled by the set containing the empty assumption set, { {} }. Assumptions are a distinguished subset of facts representing non-monotonically derived facts.

The assumptions used in the derivation of facts appear in the ATMS label of derived facts. In general, the assumption set for a derived fact, f, denoted $A(f)$, is computed by

$$A(f) = \bigcup_{i=1}^{n} A(p_i)$$

where the p_i are the antecedents that directly derive f. Assumption set $\{A1\}$ is the label for the fact Lives(Opus), and was found by taking the union of the assumption sets for nodes A1 and R3. If there are no assumption sets that support a node, represented by the empty label, { }, the fact is not believed in any reasoning context.

When a contradiction is discovered, the assumptions of the contradictory facts are unioned, creating an assumption set that is capable of supporting the derivation of an inconsistent combination of facts. This assumption set is often referred to as the "NOGOOD" assumption set, or an INCONSISTENT assumption set. The label for each node is checked to determine if any assumption sets contain the INCONSISTENT assumption set. Hence, the process of belief revision is reduced to subset operations.

In our previous example, suppose we now observe Pengiun(Opus). This gives rise to the belief ¬Flies(Opus). The problem solver makes the observation that ¬Flies(Opus) and Flies(Opus) are inconsistent facts, and notifies the ATMS of the contradiction. In response, the ATMS unions the assumption sets of the two, creating the INCONSISTENT assumption set, $\{A1, A2\}$ The ATMS then checks the label of each node to determine if it contains the INCONSISTENT assumption set, deleting any that are found. The assumption set is then cached to prevent further inferences involving the inconsistent set of facts.

Node	Fact	Label	
R1	If Bird(x) M:Flies(x) Then Flies(x)	{ {} }	Premise
R2	If Penguin(x) Then ¬ Flies(x)	{ {} }	Premise
R3	If Shoved_Into_Grand_Canyon(x) and Flies(x) Then Lives(x)	{ {} }	Premise
P1	Bird(Opus)	{ {} }	Premise
A1	Flies(Opus)	{ {A1} }	Assumption
P2	Shoved_Into_Grand_Canyon(Opus)	{ {} }	Premise
D1	Lives(Opus)	{ {A1} }	Derivation
P3	Penguin(Opus)	{ {} }	Premise
A2	¬Flies(Opus)	{ {A2} }	Assumption

5 What's In a Label

In the preceding sections we have discussed the dependency records used by each of the JTMS and ATMS in labelling problem-solver beliefs as BELIEVED or DISBELIEVED. The JTMS dependency record consists largely of the justification, which is non-monotonic, and allows us to pursue a single set of assumptions or context at a time. The ATMS dependency record contains both justification and a culmulative record of founding assumptions, and allows the problem solver to pursue multiple lines of reasoning or contexts at once. Both TMS systems support multiple justifications for a fact. In general, a number of different kinds of TMS are possible, allowing for monotonic or non-monotonic justifications, multiple or single justifications for a fact, multiple or single contexts examined at once, probabilistic or non-probabilistic belief status, and so on. These various TMS "features" define the dependency structures and the algorithms that maintain them for a particular style of TMS, and hence directly influence the computational expense of relying on a TMS-based system. The selection of a particular type of TMS should make sense with how the problem solver needs to make inferences and explore the problem space.

For example, our rule-based problem solver for signal interpretation [Mason et. al. 1988], had the following features. First, the search space was never very deep (we had no inference chains longer then 10 or so), but each

step of the inferencing was computationally intensive. Second, we wanted to find all possible interpretations, but the inferencing rarely involved more than one justification for a fact. The first implementation of our TMS, was very much like the ATMS described in [deKleer 1986]. This system was built using Zeta-lisp on a Symbolics, and it ran like a snail. By running a CPU-time profiler we observed that most of the time was spent manipulating dependency information. We simplified the dependency structure and omitted a number of assumption set optimization strategies. It now runs to our satisfaction.

In general, domains that involve little backtracking or require a single or few solutions, will pay a penalty for using an ATMS-style TMS, relying on dependency structures that were designed for problem solvers needing all or most solutions. At the same time, traditional ATMSes (using the belief updating algorithm defined in [deKleer 1986]) have complex labels, and are computationally expensive to maintain as well. The reader may realize these observations by examining the algorithms for truth maintenance in each of the JTMS and ATMS.

5.1 Truth Maintenance in the JTMS

Recall that truth maintenance (or belief updating) is the process of relabeling the support-statuses of TMS nodes so that the set of beliefs currently in the database is consistent. In the JTMS, the truth maintenance process may be invoked whenever the problem solver presents a justification to the TMS, or alternatively, whenever a contradiction is discovered. If a justification for a new node is invalid, or if the node is already present and IN, then no relabeling is necessary. If the justification changes the support-status of a node, as when a fact is re-derived with a different justification, then the node and all its consequences must be relabeled. In addition, any nodes whose OUT-list refers to a node whose status is changing from OUT to IN must be checked for relabeling as well. This process proceeds recursively for the network of dependency nodes and must be done for each node that changes belief-status until all affected nodes have been relabeled. When the process has finished, the TMS informs the problem solver of the new belief statuses. The algorithm will not terminate if unsatisfiable dependency circularities exist. For example:

node X (SL () (Y))
node Y (SL () (Z))
node Z (SL () (X))

For a more detailed description of the truth maintenance algorithm the reader is referred to [Doyle 1979] or [Charniak and McDermott 1986].

5.1.1 Dependency Directed Backtracking

The TMS relies on the problem solver to notify the TMS when beliefs are involved in contradictions. The TMS gathers the assumptions for the contradiction nodes and marks one of them, designated as the *culprit*, as OUT. The assumptions are located by recursively tracing back through the justifications for the contradiction node, finding the antecedents of antecedents, and so on, accumulating the assumption nodes that directly support the contradiction node. The TMS then creates a *nogood* node representing the notion that the combination of these assumptions is contradictory. The *nogood* node is justified by a CP justification. The *nogood* node is IN as long as the set of assumptions implies the contradiction.

The process continues by picking one of the assumption nodes in the NOGOOD set as the culprit to be relabeled *OUT* and removed from the current belief set. This is done by giving support to one of the OUT-list nodes for the culprit assumption node. The choice of a culprit assumption is arbitrary, and may result in a repetition of this process should the contradiction be derived again.

5.1.2 Truth Maintenance in the ATMS

Recalling that a node is derivable from the assumption sets contained in its label, the label of a node implicitly defines all the contexts in which the node may be a member: a node is a member of only those contexts whose characterizing environments are supersets of the environments in its label. Membership within a context may be checked by a subset test.

As with Doyle's JTMS, the problem solver must inform the ATMS of the contradiction. When a contradiction is discovered, environments that are supersets of the contradictory one are removed from all labels, leaving the database in such a form that no facts imply the contradiction. The

contradictory assumption set is then recorded so that no future inferences based on the environments known to be contradictory will be made.

These concepts are illustrated by the following example Suppose environment E contains assumptions $\{A,\ B,\ C\}$. The ATMS creates the following nodes in response to the justifications provided by the problem solver.

Justification	ATMS NODE
$a \rightarrow d$	$< d,\ \{\{A\}\},\ \ldots >$
$c \rightarrow e$	$< e,\ \{\{C\}\},\ \ldots >$
$a \wedge c \rightarrow f$	$< f,\ \{\{A,\ C\}\},\ \ldots >$
$d \wedge e \rightarrow \perp$	

The ATMS computes that $\{A,\ C\}$ is contradictory. Only fact f is affected. Environments $\{A,\ C\}$ and $\{A,\ B,\ C\}$ are contradictory, while derivations relying only on $\{A\}$, $\{C\}$, $\{A,\ B\}$, or $\{B,\ C\}$ still hold.

When a new justification for an existing belief is supplied by the problem solver, the label of a node is updated by the ATMS in the following way. The additional contexts in which the node holds are the union of the contexts of the antecedents in the new justification. The characterizing environments of the new contexts are computed by taking the union of the possible combinations of environments in the label from each of the antecedents, and then removing any labels that are supersets of nogood assumption sets. The minimal environments are then formed by removing the environments that are supersets of any other environments in the label.

An example adapted from deKleer[deKleer 86] illustrates this procedure.

node x: $< \ldots,\ \{\{A,\ B\},\ \{B,\ C,\ E\}\},\ \{\ldots\} >$
node y: $< \ldots,\ \{\{A,\ C\},\ \{D,\ E\}\},\ \{\ldots\} >$
node z: $< \ldots,\ \{\{A,\ B,\ C\}\},\ \{\} >$
nogood $\{A,\ B,\ D\}$

The problem solver deduces a new justification for z, $x \wedge y \rightarrow z$

The union of combinations of environments in x and y's labels is $\{\{A,\ B,\ C\},\ \{A,\ B,\ D,\ E\},\ \{A,\ B,\ C,\ E\},\ \{B,\ C,\ D,\ E\}\}$

Removing the contradictory $\{A,\ B,\ D,\ E\}$ and the superset $\{A,\ B,\ C,\ E\}$, the new label of node z is

$$< \ldots,\ \{\{A,\ B,\ C\},\ \{B,\ C,\ D,\ E\}\},\ \ldots >$$

The new labeling is then propagated to the consequents of the relabeled node.

6 General Comments

Historically, truth maintenance mechanisms were designed to improve problem-solving efficiency in programming languages for planning, such as Conniver [McDermott and Sussman 1972]. The basic idea was to save the fact base (working memory) at each major decision point so that it could be reinstated should the next few choices result in an inconsistency. Stallman and Sussman's programming language, ARS, for Computer-aided Circuit Analysis [Stallman and Sussman 1977], used the notion of dependency records for an intelligent cache and for dependency-directed backtracking. They also kept lists of contradictory or "NOGOOD" assertions as a technique to prevent incorrect choices from being repeated. The first presentation of these techniques as an independent programming module was in [Doyle 1979]. Although Doyle developed the TMS as a domain independent search tool, formal analysis of the resulting reasoning behavior led to the development of non-monotonic logic [McDermott and Doyle 1980].

While TMSes seem to have taken on a life in themselves, it is important to keep the idea of a TMS in perspective. Namely, it serves as an efficiency mechanism for problem solving. With or without a TMS, the same solutions must be found. However, the idea of a TMS is to relieve the problem solver from the burden of belief updating, and to do it in an efficient manner. Use of an ATMS can relieve the problem solver from task of re-deriving beliefs during search. TMSes in general, allow the problem solver to assume a contradiction-free database of beliefs.

7 Summary

This brief note is intended to familiarize the non-TMS audience with some of the basic ideas surrounding classic TMS systems, namely the JTMS and the ATMS. Topics of further interest include the relation between non-monotonic logics and TMSes, efficiency and search issues, complexity concerns, as well as the variety of TMS systems that have surfaced in the past decade or

so. These include probabilisitc-based TMS systems, fuzzy TMS systems, tri-valued belief systems, and so on.

References

[**Anderson and Belnap 1975**] A. R. Anderson and Nuel D. Belnap, Jr. *Entailment The Logic of Relevance and Necessity*, Princeton University Press, 1975.

[**Bowen and Mayhew 1989**] J. Bowen and J. Mayhew, "Consistency Maintinance in the REV-graph Environment," TR AIVRU 020, University of Sheffield, 1986.

[**Chandra 1991**] C. Chandra, "Applications of Truth Maintenance Systems to the Development of Evolutionary Software," Unpublished paper, Dept. of Computer Science, University of California, Berkeley, 1991.

[**deKleer 1986**] J. deKleer, "An Assumption Based TMS," *Artificial Intelligence* **28** (1986), 197-224.

[**deKleer and Williams 1986**] J. deKleer and B. C. Williams, "Back to backtracking: Controlling the ATMS," in *Proceedings AAAI-86*, Philadelphia, PA, (1986), 132-129.

[**deKleer and Williams 1987**] J. deKleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, 32, No. 1, 1987, 97-130.

[**Doyle 1978**] J. Doyle, "Truth Maintenance Systems for Problem Solving," Technical Report, AI-TR-419, AI Lab, MIT, Cambridge, MA, 1978.

[**Doyle 1979**] J. Doyle, "A Truth Maintenance System," *Artificial Intelligence* **12**, No. 3, November 1979, 495-516.

[**Doyle 1983**] J. Doyle, "The Ins and Outs of Reason Maintenance," in *Proceedings IJCAI-83*, Karlsrue, F.R.G., (1983),

[**Ginsberg 1987**] M. Ginsberg (Ed.), Readings in Non-Monontic Reasoning, Morgan-Kaufmann, 1987, 15.

[**Jones and Millington 1988**] J. Jones and M. Millington, "Modelling Unix Users with an Assumption-Based Truth Mantenance System: Some Preliminary Findings," in Reason Maintenance Systems and their Applications, Smith and Kelleher (Eds.), Ellis Horwood Limited, Chichester, UK. 1988, 134-154.

[**Rose and Langley 1986**] D. Rose and P. Langley, "Chemical Discovery as Belief Revision," in Machine Learning **1, Kluwer Academic Publishers, Boston, Massachusetts, 1986, 423-451.**

[Martins 1983] J. Martins, *Reasoning in Multiple Belief Spaces*, Ph.D. dissertation, Department of Computer Science, Tech Report No. 203, State University of New York at Buffalo, Buffalo, NY, 1983.

[Martins 1990] J. P. Martins, "The Truth, the Whole Truth, and Nothing But the Truth: An Indexed Bibliography to the Literature of Truth Maintenance Systems," *AI Magazine*, Fall 1990, 7-25.

[Mason et. al. 1988] C. Mason, R. Johnson, R. Searfus, D. Lager, and T. Canales, "A Seismic Event Analyzer for Nuclear Test Ban Treaty Verification" *Proc. Third Intl. Conf on Applications of Artificial Intelligence in Engineering*, August 1988, Stanford, CA.

[Mason and Johnson 1989] C. Mason and R. Johnson, "The DATMS: A Framework for Distributed Assumption-Based Reasoning," *Distributed Artificial Intelligence, Vol. II*, Eds. Gasser and Huhns, Pitman and Morgan-Kaufman Publishers, San Mateo, California , 1989.

[McAllester 1980] J. McAllester, "A Three Valued Truth Maintenance System," AI Memo 551, AI Lab, MIT, Cambridge, MA, 1980.

[McDermott and Doyle 1980] D. McDermott and J. Doyle, "Non-Monotonic
Logic I," *Artificial Intelligence*13 (April 1980), 41-72.

[McDermott and Sussman 1972] D. McDermott and G. Sussman, "The Conniver Reference Manual," MIT AI Memo No. 259a, 1972.

[Provan 1987] G. Provan, "Efficiency Analysis of Multiple-Context TMSs in Scene Representation" *Proc. Sixth National Conference on Artificial Intelligence*,1, 1987.

[Reiter 1988] R. Reiter, "Nonmonotonic Reasoning," in Exploring Artificial Intel- ligence, H. E. Shrobe and American Association for Artificial Intelligence (Eds.), Morgan-Kaufmann, San Mateo, California, 1988, 439-482.

[Reiter 1980] R. Reiter, "A Logic for Default Reasoning," *Artificial Intelligence*, 13 (April 1980), 81-132.

[Rich and Knight 1991] E. Rich and K. Knight, Introduction to Artificial Intelligence, 2nd Edition, McGraw-Hill Publishing Co., 1991.

[Stallman and Sussman 1977] R. Stallman, and G. Sussman. "Forward Reasoning and Dependency-Directed-Backtracking In a System For Computer Aided Circuit Analysis," *Artificial Intelligence* 9(2):135-196.

[Williams 1984] C. Williams, "ART the Advanced Reasoning Tool – Conceptual Overview," Inference Corp., 1984.