Consistency Reasoning in Knowledge Systems

by

Ying Kit Wong

A thesis submitted in conformity with the partial requirements

for the degree of Master of Philosophy

Department of Systems Engineering and Engineering Management

The Chinese University of Hong Kong

June, 1997

Acknowledgments

I sincerely thank my supervisor, Professor Boon Toh Low, for his guidance, encouragement, patience and kindness. I was very fortunate to have him as supervisor when I was in this department as a M.Phil student. He introduced me into the topic areas of this thesis and directed the whole course of my research. Without his kindness and support, this thesis would not have been possible. I have my greatest respect for him.

I thank my office mates, Bernard Chan, Edward Lau, Kaiser So, Louis Lo, Lung Ng and Sharon Wong and many others who made life in the Department of Systems Engineering and Engineering Management.

Finally, Iris Chan is the one who helped me pull through even the most difficult time in these years and who made everything possible. I thank her for all the encouragement and love.

Abstract

This thesis studies a consistency reasoning system for knowledge bases with a three-valued inference network called NLBN to model human commonsense reasoning. It is based on a weaker notion of commonsense consistency called W-Consistency and a consistency reasoning process. A prototype of this reasoning system is built to formalize some nonmonotonic reasoning benchmark problems and validate the consistency method used. Moreover, comparisons between NLBN with W-Consistency and other systems (AGM Logic with Epistemic Entrenchment, Truth Maintenance Systems and G-Consistency) have been made to pinpoint the differences of characteristics of these systems.

With this system, beliefs including contradictory ones, can be updated onto the belief state in any sequence and the resulting beliefs are always W-Consistent and unique. In addition to its application to commonsense nonmonotonic reasoning where non-immediate consistency maintenance of beliefs can be deferred to a suitable time at a later stage, it is also useful for merges of possibly inconsistent knowledge bases to form a set of unique and consistent knowledge.

Human beings do not only have one kind of consistency reasoning to resolve all types of inconsistencies. People may use one kind in certain situations and use another kind in other cases. Different methods of consistency

reasoning are tried to mixed or integrated in NLBN to make it more flexible and accommodative formalism to model real-world reasoning about inconsistent beliefs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ultimate goal of Artificial Intelligence is to model the way how human behave and reason. Beyond any doubt, inconsistent information does not happen accidentally in humans' daily life. However, people can still make inference reasonably. As a result, any systems aiming at modeling human reasoning should have this characteristic and it is worthwhile to investigate the adequacy of the methodology used to model it.

In this thesis, I try to capture some of the most important elements which human commonsense reasoning shows and model them in an inference network, called Neural-Logic Belief Network (NLBN) [18], with a new consistency reasoning method.

Before any discussion, let us look at an example that one may encounter when he/she was a student and then, I generalize the characteristics of hu-

man commonsense reasoning using this example.

A Student During An Examination

Suppose a student is doing a multiple choice question in the exam. There are three choices: *i*, *ii* and *iii*. After examining, he/she thinks that none of them seems to be the answer (more than one could be chosen in this question). However, He/she also believes that the solution must be the combination of them because there are no other choices in this question. At this stage, He/she holds the following beliefs in his belief state:

1. *i* is not the answer.

2. *ii* is not the answer.

3. *iii* is not the answer.

4. *i* is the answer or *ii* is the answer or *iii* is the answer.

If we use A to denote "*i* is the answer", B denotes "*ii* is the answer." and C denotes "*iii* is the answer.". The student has beliefs:

1. $\neg A$

2. $\neg B$

3. $\neg C$

4. $A \vee B \vee C$

Obviously, they are inconsistent. If he/she has time, he/she may skip this question and continue with the next one. The next question is an essay question concerned about proving a mathematical theory. When the student does this question, he/she is very cautious and makes sure that each part is valid in the proof. As time goes by, he/she must return to inconsistent beliefs about the multiple choice question. He/she usually keeps $A \vee B \vee C$ because there are no other choices and is not certain about the states of individual beliefs A, B and C. As a result, he/she decides to ask the examiner to clarify the question. After getting more information from the examiner, he/she guesses that i is a more reasonable answer and writes it down. At this time, he/she has the following belief state:

1. A

2. $\neg B$

3. $\neg C$

4. $A \vee B \vee C$

This type of reasoning process is very common during the examination sessions. This example shows some important characteristics of human commonsense reasoning which will be discussed in the next section.

1.1 Characteristics of Human Commonsense Reasoning

Before we try to model the human commonsense reasoning, we have to know what the characteristics of our commonsense reasoning are. The following five characteristics for human commonsense are assumed in this thesis:

1. **Human begins always make decisions or inference with incomplete information and it is nonmonotonic in nature.** In the real life, we are doing lots of reasoning every day. Among them, most of them are based on partial information. We cannot wait for complete information because of cost, time constraints or impossible collection of all information. What we infer based on the incomplete knowledge (or beliefs) [1] may not be the best solution. When new information comes in, our previous conclusions may be withdrawn. In the example, the student has to give an answer to the multiple choice question even though he/she does not know the exact solution.

2. **Human beings have the ability to reason in the presence of inconsistency.** Conflicting information is not rare in the world due to different sources of the same information. Beliefs derived from other knowledge or beliefs may also inconsistent if contradictory rules are used. We are used to reason in such an environment and still behave

[1] The two terms *"knowledge"* and *"belief"* are used interchangeably in this thesis.

rationally. The student in the example has an inconsistent belief state at some time: $\{ \neg A, \neg B, \neg C, A \vee B \vee C \}$ and still can finish other questions.

3. **Human beings do not rely on a particular consistency reasoning method to deal with all problems. It is free to change from one kind to another to fit the situation.** We do not only have one kind of consistency reasoning to resolve any inconsistent beliefs. We may use one kind in a certain situation or environment and use another in other cases. This gives us a flexible and accommodative capability to deal with our daily lives. For example, we use a strict consistency reasoning when solving an mathematical equation while use a more relax one to decide where to go for lunch. In the example, while the student holds contradictory beliefs in one question, he/she uses a strict consistency type when proving a mathematical theory.

4. **Human beings have different degrees of credibility to different knowledge. If contradictory beliefs exist, the stronger one suppresses the weaker one and no beliefs are removed in this reasoning process.** When the student guesses the solution, he/she writes down the one which he/she thinks it is the most likely the answer. Therefore, we do not believe all kinds of information in the same degree. Different sources have different credibility. Information from an radar is usually more credible than that from our eyes. If two beliefs of the

same fact are contradictory, we generally agree that the stronger belief can over-ride the weaker one and we can reason based on the stronger belief. Moreover, the weaker belief does not discard from our mind and it is only suppressed. We can always retrieve this suppressed belief if we want to.

5. **Human beings can resolve any inconsistency at any time depending on situations.** It is plausible that human beings do not always maintain consistency of their belief states immediately in view of a belief change. This observation is obvious when the belief change is not of immediate interest to that person. For instances, any unimportant observations in daily life are just being recorded, and no proper consistency reasoning is carried out for each and every piece of information accepted or rejected. They will be used for reasoning only when a need arises. Whether they are consistent with others in the belief state is then examined to satisfy the reasoning requirement. This characteristic is obvious in the example, the student only tries to resolve the inconsistency in the multiple choice question when he/she is required to do so. This type of *lazy consistency reasoning* is the fifth elements of human commonsense reasoning.

The above characteristics are some of the most important elements in human commonsense reasoning. Studies on how to model consistency reasoning of knowledge systems throughout this thesis are based on these elements.

1.2 Neural-Logic Belief Network as the Basic Inconsistency Reasoning System

There are two main concerns in constructing a knowledge system for modeling human reasoning. The first one is the problem of formalization, that is, how to translate the knowledge of human beings into a format that can be stored, retrieved and manipulated efficiently. The second concern is about the inconsistency handling and how the system deals with the changes of knowledge base when new information comes in.

In this thesis, we concentrate on the second issue of inconsistency handling based on the assumption that we have chosen *Neural-Logic Belief Network* (NLBN) [18] as our basic system. There are three main paradigms of representation of knowledge in AI field. The logic paradigm uses various forms of logics, typically classical propositional or first-order logics, two-valued or three-valued. For example, [5, 22, 26] are some of them in this paradigm which has the advantage of having well-defined semantics, well-studied inference algorithms. The connectionist paradigm recognizes that knowledge are distributive, competitive and co-operative in nature, and they takes on the distributed and parallel activity in a network [27, 23]. The third paradigm tries to exploit the relative strengths of each paradigm and combine them to improve performance. (e.g. [29])

NLBN is hybrid approach of two paradigms. It is a three-valued, acyclical directed graph consisting of nodes and directed links from one node to another with a neural-net computation model. It uses computational functions to propagate node values to all nodes while each node and link has its semantic interpretation so that the symbolic knowledge is represented explicitly.

NLBN can represent basic logical connectives of the Kleene's strong three-valued logic [11], IF-THEN rule used in commonsense reasoning and human-biased relations [19]. Moreover, in Chapter 8, we show that different consistency reasoning methods can be integrated or mixed in a NLBN with some modification of computation functions (if needed).

In Chapter 2 of this thesis, I will briefly summarize the basic structure of a NLBN and the definitions of the computation functions for propagating node values to other nodes.

1.3 Consistency of Knowledge

Consistency of knowledge is one of the most important consideration in knowledge representation and reasoning systems. Classical logical consistency is the common starting point for logic based formalisms [7]. One well-known dynamic belief revision system is the AGM logic with Epistemic Entrenchment [5] in which consistency of the closures of theories is one of the

fundamental criteria. In Default Reasoning [26] and Circumscriptive Logics [14], extensions of theories have to be logically consistent . In these systems, contradictory information is not allowed and has to be expelled form the final belief state or everything can be deduced in the presence of inconsistency.

In recent decades, many researchers have recognized this characteristic undesirable [1, 14, 8]. The mismatch between classical logics and human reasoning logics have led to non-classical logical consistent formalisms for commonsense reasoning. Representations include multi-valued logic (e.g. [1, 14, 10, 15]), paraconsistent logic (e.g. [28, 25, 4]), and inference network (e.g. [19]). In [1], a four-valued logic is suggested to be used as an actual guide to reasoning and handling inconsistency while [14] employed three-valued logic to construct their systems. *Annotated Predicate Calculus* is proposed in [10] in which different strength default truth values are used and *epistemic inconsistency* is used to represent conflicting beliefs of a reasoner which should be tolerated. In [15], a logic is defined that allows an agent to reason consistently, even though there are inconsistenies in the belief state. Paraconsistent logic avoids the triviality and allows us to reason in the presence of inconsistency. In [25], the *logic of paradox LP* is introduced which can localize contradictions although some inferences that are classically valid are not valid. As a result, the *logic of minimal paradox* LP_m [4] provides a nonmonotonic extension of LP and overcomes the weakness of LP. In [19], three-valued node values are used in the inference network. It accepts contra-

dictory information and has the ability to resolve inconsistency in the belief state. All these systems model realistic notions that appear commonly in human reasoning such as "*contradiction*", "*indeterminate*" and "*unknown*" in different extends.

In Chapter 3, I define a *Weak Notion of Consistency*, W-Consistency which is a weaker and more general than the strict classical logical consistency. Under the definition of W-Consistency, a *W-Consistency reasoning process* [20] can be built up which is used to resolve any inconsistency if present. The basic principle of the consistency reasoning process is to maintain as many beliefs unchanged as possible and the inconsistent beliefs are *suppressed* instead of expelling from the final belief state. This process is carried out when there is an update operation on the NLBN and it must terminate finitely with a consistent belief state.

1.4 Update Sequence Independence in Belief States

Although many systems have been constructed to resolve any inconsistency in their own ways, most of them are update sequence dependent. That means that the final state of the knowledge base is dependent on the sequence of

the update operations even though the same operations are used. For example, [5] did not mention the problem of update sequence in AGM Logic so that it is difficult to model the *iterated belief change* [24]. [19] defined the G-Consistency and under the definition, the consistency reasoning process must be carried out immediately when a belief change arises. Different sequences of update operations provide possible variations of G-consistent belief states.

In many practical situation, update sequence dependence is not desirable in many situations. For example, when merging multiple knowledge bases, we always want to obtain a unique final knowledge base and should not concern about the sequence of which two are merged first or all must be merged at the same time. For a database management system, the update operations are random and unpredictable. If the system is sequence dependent, the final result is based on the ordering of the update operations.

In Chapter 7, we use an example to show that AGM Logic with Epistemic Entrenchment and G-Consistency with NLBN is update sequence dependent. On the other hand, the proposed W-Consistency is truly update sequence independent. No matter how we change the ordering of operations, the same final belief state can be obtained.

1.5 Lazy Consistency Reasoning

Large amount of update operations may be applied to a knowledge system in a short interval. The consistency reasoning process must be applied intensively to ensure the whole knowledge set is consistent. However, consistency reasoning is a potentially time and resource consuming task. The efficiency of the knowledge system may be slowed down or some important updates are delayed. It is better if the consistency reasoning process can be applied at any time after arbitrary numbers of updates. This brings the concern of *lazy consistency reasoning.*

Moreover, in human commonsense reasoning, people do not always immediately maintain consistencies of our belief states in view of a belief change. This observation is true when the newly accepted or rejected belief is not of immediate interest to the person. One possibility is that, given a very important belief change or when the person is rational, a consistency reasoning is carried out to accommodate the new belief change when there is sufficient time and resource for the task after the immediate reasoning needs to carry out normal functions of life are satisfied. Obviously, minor observations in our daily life are just being registered, and no proper consistency reasoning is carried out for each and every piece of information accepted or rejected. They will be used for reasoning only when a need arises. Whether they are consistent with other belief state are then examined and a consistency rea-

soning process is carried out to satisfy the reasoning requirement. This type of *lazy consistency reasoning* is common in human cognition. As a result, it is necessary to incorporate this kind of characteristic into a knowledge representation and reasoning system to model the human commonsense reasoning.

W-Consistency used in NLBN can be used to model this kind of characteristic. How lazy (late) we evaluate the belief state does not affect the uniqueness of final state. Combined with the result of update sequence independent characteristics, W-Consistency can be freely used at any stage and the uniqueness of final belief state is preserved. Besides the theoretical result, an implementation of NLBN, called *XHOPES* is described in Chapter 4 which can be used to test the validity of operations and consistency reasoning process in NLBN.

One application of lazy consistency reasoning is merging multiple arbitrary inconsistent knowledge bases into a consistent final one. Those knowledge bases being merged can be updated individually and we do not need to concern about the consistency of each one if they are not used individually or any decisions made using these inconsistency knowledge bases may not have drastic effect. When a final knowledge base is needed by combining different ones, a Lazy W-Consistency reasoning is applied. This final knowledge base is W-Consistent and unique based on a specific set of update operations.

1.6 Comparison of W-Consistency with Other Systems

In Chapter 5 and 6, I compare NLBN using W-Consistency with other systems and show the differences in terms of characteristics. The first system I compare with is AGM Logic with Epistemic Entrenchment [5]. It is one of the most influential systems for belief revisions. There are three operations that can operate on a belief state: *expansion*, *revision* and *contraction*. It proposes several postulates for each operation based on the principle of *information economy* [5]. One of the constructive ways to build a system conforming with the postulates is using epistemic entrenchment which states that knowledge that are accepted in a given belief set have different degrees of epistemic entrenchment. When a belief set is revised or contracted, the beliefs that are given up are those having the lowest degrees of epistemic entrenchment. NLBN with W-Consistency has similar principle when a belief state is changed. It keeps as many beliefs as possible and using the *degree-of-beliefs* values to guide the operations. However, NLBN with W-Consistency does not follow all postulates that AGM Logic supports. The main reason is that NLBN with W-Consistency can accept contradictory information in the same belief state while AGM Logic cannot.

The second comparison is with G-Consistency [19]. Both G-Consistency and W-Consistency are developed for NLBN and they are weaker version than

the classical logical consistency. However, G-Consistency is more decisive for handling contradictory information. It tries to find out more reasonable beliefs among contradictory ones and continues to use them for reasoning. In contrast, W-Consistency defers the reasoning based on the contradictory beliefs until more information is collected.

Another system with different characteristics chosen for comparison with W-Consistency is Truth Maintenance System (TMS) [3]. TMS is part of the reasoning system that works with a *problem solver* (PS) to maintain consistency of the belief state and remove contradictions. Each belief in TMS can be *in* of *out*. However, positive and negative view of the same belief (e.g. p and $\neg p$) have to be represented separately so that there are four combinations of states for a belief. It also recognize the importance of minimal change to existing beliefs when updating the current set of beliefs. It is different to NLBN in that it removes contradictory beliefs instead of suppressing them and it may inherits the risks of running into loops during truth maintenance.

1.7 Integration of Different Methods in One Formalization

Human begins do not only use one kind of consistency reasoning method. We can make use of NLBN to model this kind of characteristic by integrating different methods of inconsistency handling under the same framework. In Chapter 8, I mix G-Consistency and W-Consistency together and study the behavior of the mixed NLBN. At first, both of the two consistencies can easily integrated because of using similar formalism. There is a danger that the mixed final belief state may not be unique due to the sequence dependence of G-Consistency. However, given a specific treatment of consistency reasoning in NLBN, we can guarantee that some portions of NLBN can preserve the uniqueness of the belief state which is fully discussed in Chapter 8.

On the other hand, NLBN is also used to model the way the *Truth Maintenance Systems* (TMS) handling inconsistency [21]. It is shown that by mapping the states in TMS into node values of NLBN, and, *SL-justifications* and *CP-justifications* into suitable IF-THEN rules. NLBN can reason similarly to TMS's way. It is a step further to integrate different methods into a unify platform such as NLBN so that each method can be applied to different situations.

Chapter 2

Neural-Logic Belief Network (NLBN)

2.1 Definitions

A *Neural-Logic Belief Network* (NLBN) [18] is a general belief inference network revision system. It is an acyclical directed graph consisting of nodes and directed links from one node to another with a neural-net like computation model. Figure 2.1 shows an example of a NLBN.

There are two types of nodes: *input nodes* and *base nodes.* Input nodes receive input beliefs and propagate them via directed input links to the relevant base nodes. Each node represents a *proposition* and has a *node value* to indicate its current belief state. To model incomplete and inconsistent input knowledge, different input nodes are used to represent different views of the

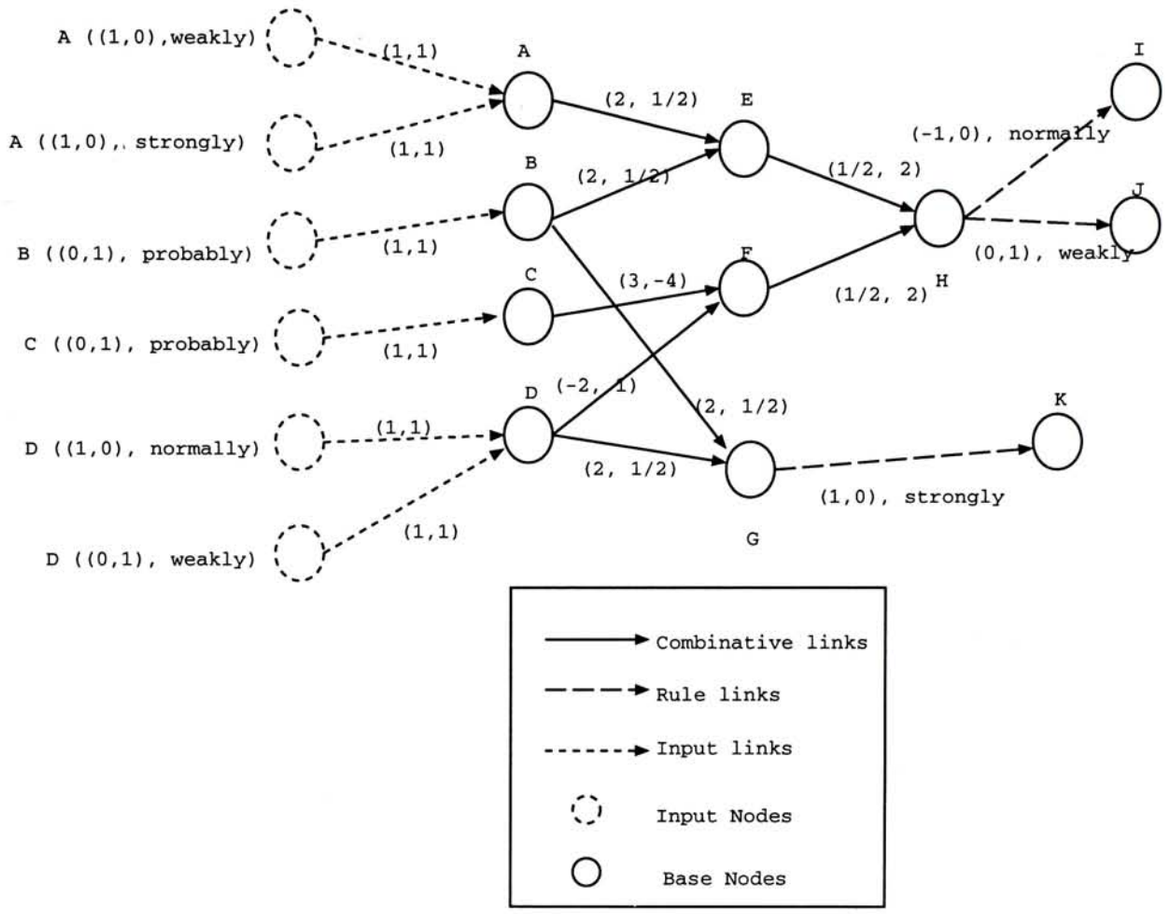


Figure 2.1: General Structure of a NLBN

same proposition from different sources. But for base nodes, each proposition is uniquely represented by one node (only either the positive view or negative view is represented). The set of all propositions of base nodes, B, is called the *belief base* and a *belief state* S is a collection of all propositions in positive and/or negative form given a belief base B.

Each *node value* is an ordered pair: (*proposition-value, degree-of-belief value*) and there are three possibilities for the *proposition-value* (t, f):

- (1, 0) means that the proposition associated with the node is believed;

- (0, 1) that the negation of the proposition is believed;
- (0, 0) that it is neither believed nor disbelieved.

The degree-of-belief values, written as deg[(The proposition of the node)], represent the belief strength of the propositions concerned. This induces a total asymmetric order (TAO) on all the degree-of-belief values in a NLBN. For example, we may have a set of beliefs with the following order:

$$0 < \textit{weakly} < \textit{probably} < \textit{normally} < \textit{strongly} < \textit{true}$$

A base node a having a node value of ((0, 0), deg[a]) means that it is uncertain. When deg[a]>0 means that this belief currently has contradictory inputs of the same strength as indicated by its degree-of-belief value deg[a]. That is both $a \in S$ and $\neg a \in S$. A special case is ((0, 0), 0) where the degree is the weakest value, that is, $a \notin S$ and $\neg a \notin S$. This is the default node value. A base node a have a node value of ((1, 0), deg[a]) means the proposition a is believed in the belief state S, i.e. $a \in S$ and $\neg a \notin S$; if its node value is ((0, 1),deg[a]) then $\neg a \in S$ and $a \notin S$. Node values of ((1,0),0) and ((0,1),0) do not have proper semantic interpretation and they are invalid. The belief states of nodes in a NLBN and their corresponding node values can be interpreted as follows:

Belief State of node p	Node Values
t (true)	((1,0), deg> 0)
f (false)	((0,1), deg> 0)
u (uncertain)	((0,0), deg)

2.2 Computation Functions

Each directed link in the belief network is associated with an ordered pair of link weights (u, v), where $u, v \in \{\text{Real Numbers}\}$. Analogous to proposition-values, the first weight u is an excitatory value and the second weight v is an inhibitory value. There are three types of links: *combinative-links* are used to construct logical relations , *input-links* are used to propagate external input values from input nodes to their respective base nodes with the same proposition and *rule-links* are used to construct IF-THEN rules.

Typically, part of the *combined input value*: the *combined input proposition value* (t_o, f_o) to a logical expression base node o with inputs from n sub-expressions (o_1, o_2, ..., o_n, where $0 < i < n$) via n combinative links is shown in Figure 2.2.

In the Figure, each sub-expression has a proposition-value of (t_i, f_i) and a link weight of (u_i, v_i) to o. The node value of o is given by the following *Thresholding Function*:

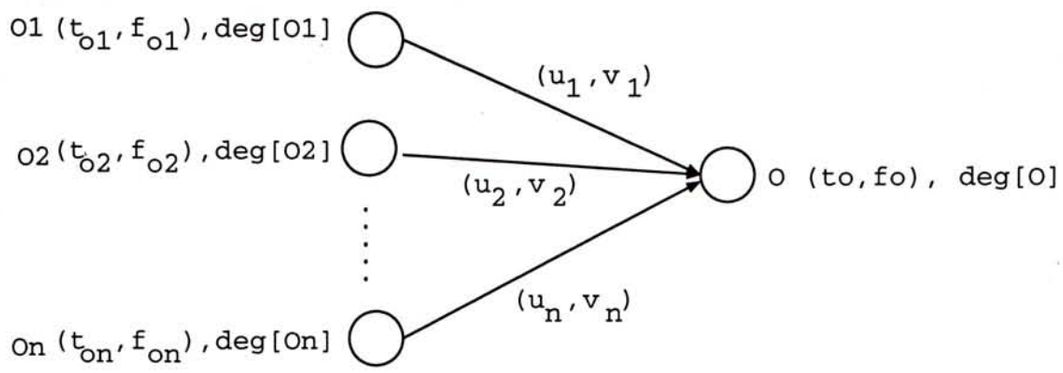


Figure 2.2: The combined input value of a logical expression

$$(t_o, f_o) = \begin{cases} (1,0) & \text{if } NET \geq 1 \\ (0,1) & \text{if } NET \leq -1 \\ (0,0) & \text{otherwise} \end{cases}$$

where the value of NET is given by the following *Transfer Function* :

$$NET = (\textstyle\sum_{i=1}^{n} E_{o_i}) - (\textstyle\sum_{i=1}^{n} I_{o_i})$$

where $E_{o_i} \in \mathbf{E}$, which is the set of excitatory inputs (i.e. inputs for the proposition o):

$$\mathbf{E} = \left\{ E_{o_i} \middle| \begin{array}{l} E_{o_i} = |f_i \times v_i| \text{ where } f_i \times v_i < 0, \text{ otherwise} \\ E_{o_i} = t_i \times u_i \text{ where } t_i \times u_i \geq 0, \text{ for all } 0 < i \leq n) \end{array} \right\}$$

and $I_{o_i} \in \mathbf{I}$, which is the set of inhibitory inputs (i.e. inputs against the proposition o):

$$\mathbf{I} = \left\{ I_{o_i} \middle| \begin{array}{l} I_{o_i} = |t_i \times u_i| \text{ where } t_i \times u_i < 0, \text{ otherwise} \\ I_{o_i} = f_i \times v_i \text{ where } f_i \times v_i \geq 0, \text{ for all } 0 < i \leq n) \end{array} \right\}$$

The other part of the *combined input value*: the *input degree-of-belief value* for node o, deg[o], depends on the computed combined input proposition value:

- If $(t_o, f_o) = (0, 0)$ then

$$deg[o] = max(deg[o_i]) \text{ for all } 0 < i \leq n \text{ where } t_{oi} = 0 \text{ and } f_{oi} = 0$$

$max(deg[o_i])$ returns the highest degree-of-belief value according to the TAO for the beliefs.

- If $(t_o, f_o) = (1, 0)$ then

$$deg[o] = \begin{cases} deg[o_k] & \text{if } E_{o_k} > 1, deg[o_k] \geq deg[o_i] \text{ for all } 0 < i \leq n, \\ & E_{o_k}, E_{o_i} \in \mathbf{E} \text{ and } i \neq k, \text{ otherwise} \\ deg[o_j] & \text{if } min[(\sum_{j=1}^{k} E_{o_j}) \geq 1], deg[o_j] \geq deg[o_i], E_{o_j} \neq 0 \\ & E_{o_i} \neq 0, \\ & E_{o_j}, E_{o_i} \in \mathbf{E}, \text{ for all } 0 < i \leq n \text{ where } i \neq j. \end{cases}$$

where $min[(\sum_{j=1}^{k} E_{o_j}) \geq 1]$ is the minimum value of the sum $\sum_{j=1}^{k} E_{o_j}$ (with a distinct value of j) for all non-zero E_{o_j} that is greater or equal to 1 (the threshold) summed by descending order of the corresponding $deg[o_j]$, in which $deg[o_j] \geq deg[o_{j+1}]$.

- If $(t_o, f_o) = (0, 1)$ then

$$deg[o] = \begin{cases} deg[o_k] & \text{if } I_{o_k} > 1, deg[o_k] \geq deg[o_i] \text{ for all } 0 < i \leq n, \\ & I_{o_k}, I_{o_i} \in \mathbf{I} \text{ and } i \neq k, \text{ otherwise} \\ deg[o_j] & \text{if } min[(\sum_{j=1}^{k} I_{o_j}) \geq 1], deg[o_j] \geq deg[o_i], I_{o_j} \neq 0, \\ & I_{o_i} \neq 0, \\ & I_{o_j}, I_{o_i} \in \mathbf{I}, \text{ for all } 0 < i \leq n \text{ where } i \neq j. \end{cases}$$

where $min[(\sum_{j=1}^{k} I_{o_j}) \geq 1]$ is defined in the similar way above.

Input-links, and rule-links are "alternative" links. The values propagated by these links are not combined with other links. They are instead considered as possible *alternative input values* to the node concerned and are each computed using the same computation functions for combinative-links. The input node value received by a base node o (whose node value is ((t_o, f_o), deg[o])) from another base node p with a proposition-value (t_p, f_p) via an alternative-link with link-weights of (u_p, v_p) and the link degree-of-belief value of deg[link] is given by:

$$(t_o, f_o) = \begin{cases} (1, 0) & \text{if } (t_p \times u_p - f_p \times v_p) \geq 1 \\ (0, 1) & \text{if } (t_p \times u_p - f_p \times v_p) \leq -1 \\ (0, 0) & \text{otherwise} \end{cases}$$

$$deg[o] = \begin{cases} deg[p] & \text{if deg}[p] \leq \text{deg}[link] \text{ and } (t_o, f_o) \neq (0, 0) \\ deg[link] & \text{if deg}[p] > \text{deg}[link] \text{ and } (t_o, f_o) \neq (0, 0) \\ 0 & \text{otherwise} \end{cases}$$

When there are n inputs from n alternative-links to a base node o, the network values from each input $((t_i, f_i), \deg[i]),\ 0 < i < n$ is first computed according to the above computation functions, then the subsumptive *selection function* below is used to pick up the strongest input among these alternatives as the computed node value.

$$(t_o, f_o) = \begin{cases} (t_k, f_k) & \text{if } \deg[k] > \deg[i],\ 0 < k \leq n,\ k \neq i, \text{ for all } 0 < i \leq n \\ (t_j, f_j) & \text{if } (t_j, f_j) = (t_k, f_k),\ \deg[j] = \deg[k], \text{ for some } j, k, 0 < j, k \leq n, \\ & j \neq k,\ \deg[j] > \deg[i],\ j \neq i,\ k \neq i, \text{ for all } 0 \leq i \leq n \\ (0, 0) & \text{otherwise} \end{cases}$$

$$deg[o] = \begin{cases} deg[k] & \text{if } \deg[k] \geq \deg[i],\ 0 < k \leq n, k \neq i, \text{ for all } 0 < i \leq n \\ 0 & \text{otherwise} \end{cases}$$

Example: The Figure 2.3 below shows an example segment of a NLBN.

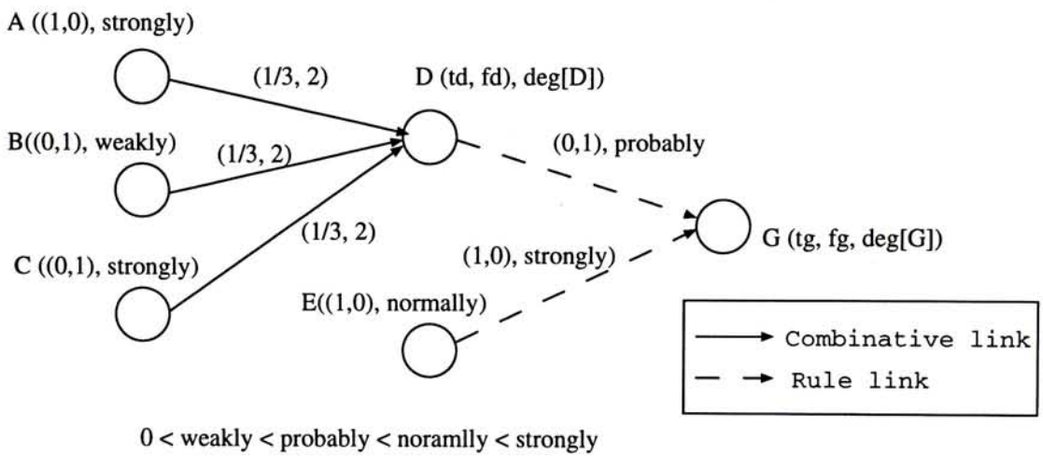


Figure 2.3: Example

Node value of D is computed using the computation function for combinative links as follow:

$$E = E_A + E_B + E_C = 1 \times \tfrac{1}{3} + 0 \times \tfrac{1}{3} + 0 \times \tfrac{1}{3} = \tfrac{1}{3}$$

$$I = I_A + I_B + I_C = 0 \times 2 + 1 \times 2 + 1 \times 2 = 4$$

$$NET = E - I = \tfrac{1}{2} - 4 = -3\tfrac{1}{3}$$

Therefore, the proposition-value of node D is (0,1).

Since $deg[C] > deg[B]$, and $I_C = 2 > 1$, $deg[D] = deg[C] = strongly$.

The node value of G is computed using computation functions for alternative links. There are two rule-links: one from D and the other one from E. Node value of D is $((0,1)$, *strongly*) and link weight is $((0,1)$, *probably*), therefore, the input node value from D is $((0,1)$, *probably*) since $deg[link] < deg[D]$ and

$$(t_D \times u_D - f_D \times v_D) = (0 \times 0) - (1 \times 1) = -1$$

and input node value from E is $((1,0)$, *normally*) since $deg[E] < deg[link]$ and

$$(t_E \times u_E - f_E \times v_E) = (1 \times 1) - (0 \times 0) = 1$$

Using the selection function for different inputs from alternative links, the stronger input node value from E becomes the node value of G: $((1,0)$, *normally*).

All strong three-value logical expressions [11] such as AND ($\wedge$), OR ($\vee$) and NOT ($\neg$) in Conjunctive Normal Form (CNF) can be constructed by *combinative-links* [18]. These are the basic logical connectives for a NLBN and all logical relations are expressed in CNF. For example, a 2-input OR $a \vee b$ is represented as shown in Figure 2.4 while a 2-input AND $a \wedge b$ is represented as shown in Figure 2.5.

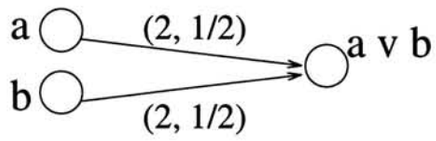


Figure 2.4: OR relation represented in a NLBN

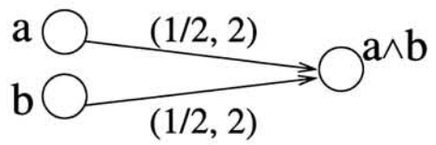
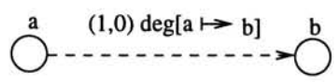


Figure 2.5: AND relation represented in a NLBN

Other than logical relations, non-classical logical relations (relations that may give rise to nonmonotonic characteristics) such as the defeasible IF-THEN rules represented by rule links can also be constructed in NLBN. This IF-THEN rule is semantically weaker than conditionals expressed by material implications ($\rightarrow$). A rule "IF a THEN b (certainty of the rule)", written as $a \mapsto b$ ($deg[a \mapsto b]$), means if we believe a, then we can derive b

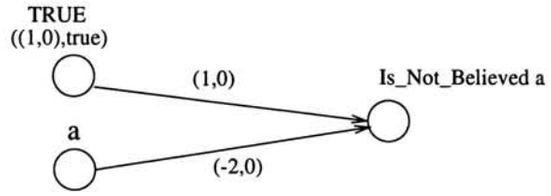
with respect to the rule degree-of-belief value. The relation of IF-THEN rule is represented in NLBN shown in Fig. 2.6.



a	t	f	u
b	t	u	u

Figure 2.6: IF-THEN rule in a NLBN

NLBN uses an **Is_Not_Believed** relation to map any given base node into two-valued modes [16]. For any given base node a, we introduce a new base node 'Is_Not_Believed a' into NLBN. The characteristic of this relation is given in Fig 2.7.



Proposition value of a	Node values of Is_Not_Believed a
(1,0)	((0,1), deg[a])
(0,0)	((1,0), true)
(0,1)	((1,0), true)

Figure 2.7: Is_Not_Believed relation

There is a set of six *Network Update Operators* for the belief network [18]: *Add*, *Update*, *Remove*, *Forget*, *Not-Conclude* and *Revise*. The operators *Add* and *Update* are used to include new beliefs into or update existing beliefs of a belief state. *Remove* and *Forget* are used to expel existing beliefs, and

Not-Conclude is used to temporarily suppress a belief to a proposition-value of (0, 0). The other operator *Revise* is for making revision to the belief state according to a new belief and it is based on the *Add* and *Remove* operators.

Chapter 3

W-Consistency Reasoning

In this chapter, we define a new kind of consistency, which is weaker than classical 3-valued logics called, W-Consistency. We adopted the following principles for the *W-Consistency Reasoning Process* [20] in NLBN:

1. Minimal change to existing beliefs—This will ensure that under the conditions of the reasoning process, the least possible number of beliefs are affected when resolving inconsistencies.

2. Suppression of beliefs—The weakest inconsistent beliefs affected in consistency reasoning are suppressed by their strongest counterparts and they remain as beliefs rather than being expelled from the beliefs.

Under the definitions described in the following subsections and the above two principles, the main *W-Consistency Reasoning Process* is formalized using the two sub-processes: *Consistency Check* (Section 3.3) and *Consistency Maintenance* (Section 3.4).

3.1 W-Consistency

Before we introduce W-Consistency, we have to define some terminologies which will be used later. Given a logical expression base node Ψ with n sub-expressions, any of its sub-expressions α_i ($0< i \leq n$) with a proposition-value (t_i, f_i) and a link weight of (u_i, v_i) has an input from α_i to Ψ, (x, y) = ($t_i \times u_i$, $f_i \times v_i$). This input is a contributing agreeing input where

$x > 0$ or $y < 0$ if the proposition-value of Ψ is (1,0); or,

$x < 0$ or $y > 0$ if the proposition-value of Ψ is (0,1).

This input is a contributing disagreeing input where

$x < 0$ or $y > 0$ if the proposition-value of Ψ is (1,0); or,

$x > 0$ or $y < 0$ if the proposition-value of Ψ is (0,1).

This input is an influencing input where

$x \neq 0$ or $y \neq 0$ when the proposition-value of Ψ is (0,0).

A sub-expression *agrees* with a logical expression if and only if it is a contributing agreeing input to the proposition-value of the logical expression. It *disagrees* with a logical expression if and only if it is a contributing disagreeing input to the proposition-value of the logical expression. Disagreeing sub-expressions are divided into *strongly disagreeing* sub-expressions and *weakly disagreeing* sub-expressions. Strongly disagreeing sub-expressions are those with degree-of-belief value greater than that of the logical expression

while weakly disagreeing sub-expressions are smaller than or equal to the logical expression. It is a *interfering* sub-expression if and only if it is a influencing input to the proposition-value of the logical expression.

A set of sub-expressions is said to be *non-interfering* with its logical expression if and only if both the combined input proposition value of this set and the proposition-value of the logical expression are (0,0):

The proposition-value of the logical expression	The combined input proposition value from its sub-expressions
(0,0)	(0,0)

A logical expression is said to be *supported* by a set of its sub-expressions if and only if the computed combined input proposition value from these sub-expressions is not (0,0) and it has the same proposition-value as the logical expression as shown below:

The proposition-value of the logical expression	The combined input proposition value from its sub-expressions
(1,0)	(1,0)
(0,1)	(0,1)

Otherwise, it is a set of non-supporting sub-expressions. For example, $a \vee b$ ((1,0), deg[$a \vee b$]) is supported by a ((1,0), deg[a]), or b ((1,0), deg[b]). $a \wedge b$ ((1,0), deg[$a \wedge b$]) is supported by a ((1,0), deg[a]) and b ((1,0), deg[b]).

A logical expression is *opposed* by a set of its sub-expressions if and only if the combined input proposition value of the set of sub-expressions is neither (0,0) nor supporting the logical expression:

The proposition-value of the logical expression	The combined input proposition value computed from a set of sub-expressions
(1,0)	(0,1)
(0,1)	(1,0)
(0,0)	(1,0)
(0,0)	(0,1)

For example, the belief $a \vee b$ ((1,0), deg[$a \vee b$]) is opposed by a ((0,1), deg[a]) and b ((0,1), deg[b]).

Definition W-Consistency A logical expression is *W-Consistent* if and only if

1. the proposition value of the logical expression is supported by the combined input proposition value, or

2. there are no weakly disagreeing sub-expressions and no interfering sub-expressions for the logical expression.

Any inconsistency is resolved by suppressing all of the weakly disagreeing sub-expressions and influencing sub-expressions to a proposition-value of (0,0) so that it is W-consistent.

3.2 Logical Suppression

Suppressions come naturally with the computation functions of Neural-Logic Belief Networks. Any proposition associated with a node in a belief network can receive contradictory inputs from different sources. Due to the competitive and co-operative nature of the computation model employed (Chapter 2), only the strongest or un-opposed inputs are selected at each base node locally to represent their current beliefs in the belief state. Although the weaker inconsistent inputs remain in the beliefs, they are just temporarily suppressed by their stronger counterparts. For example, if there are 2 contradictory inputs $((1,0)$, *strongly*$)$ and $((0,1)$, *weakly*$)$ for a belief a, the belief state of a is the stronger input $(1,0)$. The weaker input $(0,1)$ is suppressed. In consistency reasoning, a similar minimal change principle is adopted where weaker inconsistent beliefs are suppressed rather than expelled from the beliefs.

As the final belief of a logical expression (which may receive both direct inputs and a combined input from its sub-expressions) in the belief state is handled by the computation functions of belief networks locally, the only possibility where inconsistency can occur is when the input value from a sub-expression is inconsistent and weaker than or equal to the logical expression, and the current node value of the logical expression is derived from other stronger direct input(s). Thus, to maintain consistency, suppressions are re-

quired from the logical expressions to their inconsistent sub-expressions, and they are called *Logical Suppressions* [19]. A logical suppression from a logical expression making a sub-expression to a proposition-value of either (1,0) or (0,1) will turn it into an supporting input to the logical expression. This is called the *Supporting Logical Suppression.* A logical suppression from a logical expression making a sub-expression to a proposition-value of (0,0) is called a *Non-Supporting Logical Suppression.* This is carried out by introducing an opposite input, i.e. (1,0) for (0,1) or (0,1) for (1,0) to the inconsistent belief with the same degree-of-belief value of that inconsistent belief. This is the only form of logical expression required to maintain W-consistency and only one suppression per sub-expression is permissible.

Given a logical expression Ψ and if its sub-expression α (with a proposition-value of either (1,0) or (0,1)) is logically suppressed by a non-supporting suppression to β (with a proposition-value of (0,0)) and a degree-of-belief value of deg[α], where α and β are the same proposition with different proposition values (e.g. a and $\neg a$), is denoted as:

$$\alpha \overset{\Psi}{\Longrightarrow} (\beta, \deg[\alpha])$$

Since only one suppression per sub-expression is allowed for each logical expression, the maximum number of logical suppressions that can be imposed from a logical expression is equal to the number of its sub-expressions. When all possible logical suppressions have been imposed and no more suppression

is possible because others are strongly disagreeing sub-expressions, it is say to be *exhaustively suppressed.* A special case is when all the sub-expressions of a logical expression are logically suppressed, it is said to have *full suppression* or it is *fully suppressed.* A fully suppressed belief is W-consistent. If a belief is removed from the belief network, all its logical suppressions are also removed.

3.3 Consistency Check

Consistency Check is a checking process to find out inconsistent logical expression(s) in the belief state. For each logical expression concerned, a consistency check to it also extends to all its logical expressions that are *related* to each of the sub-expressions. This includes all logical expressions formed in-part by the sub-expression or its negations which can be tracked via the combination-links from the sub-expression. For example, given a sub-expression a, its related logical relations are any logical relations containing a or $\neg a$ (e.g. $a \vee b$, $(\neg a \vee c) \wedge d$, etc.).

3.4 Consistency Maintenance

Consistency Maintenance is a set of procedures for maintaining consistency of belief states. These procedures do not make any changes to the beliefs if the belief state is consistent. When a logical expression is found to be inconsistent, this set of procedures forces changes to make the belief state

Case	a b	$a \vee b$		Remarks
1	(1,0) (1,0)	(1,0)	*	*: W-consistent (logically consistent).
2	(1,0) (0,0)	(1,0)	*	†: may be W-inconsistent—suppress all
3	(1,0) (0,1)	(1,0)	*	weakly disagreeing sub-expression(s) to
4	(0,0) (1,0)	(1,0)	*	(0,0).
5	(0,0) (0,0)	(1,0)	*	§: W-inconsistent—suppress all weakly
6	(0,0) (0,1)	(1,0)	†	disagreeing and interfering sub-express-
7	(0,1) (1,0)	(1,0)	*	ion(s) to (0,0).
8	(0,1) (0,0)	(1,0)	†	#: not applicable.
9	(0,1) (0,1)	(1,0)	§	
10	(1,0) (1,0)	(0,0)	§	
11	(1,0) (0,0)	(0,0)	§	
12	(1,0) (0,1)	(0,0)	§	
13	(0,0) (1,0)	(0,0)	§	
14	(0,0) (0,0)	(0,0)	*	
15	(0,0) (0,1)	(0,0)	†	
16	(0,1) (1,0)	(0,0)	§	
17	(0,1) (0,0)	(0,0)	†	
18	(0,1) (0,1)	(0,0)	§	
19	(1,0) (1,0)	(0,1)	#	
20	(1,0) (0,0)	(0,1)	#	
21	(1,0) (0,1)	(0,1)	#	
22	(0,0) (1,0)	(0,1)	#	
23	(0,0) (0,0)	(0,1)	#	
24	(0,0) (0,1)	(0,1)	#	
25	(0,1) (1,0)	(0,1)	#	
26	(0,1) (0,0)	(0,1)	#	
27	(0,1) (0,1)	(0,1)	*	

Table 3.1: All possible case for a two input OR relation $a \vee b$

consistent by introducing appropriate logical suppression(s).

As NLBN only accepts logical expressions in conjunctive normal form, only logical expressions formed by the two basic logical connectives **AND** ($\wedge$) and **OR** ($\vee$) (together with **NOT** ($\neg$))need to be examined in consistency maintenance. To begin, let us look at a two-input disjunctive expression $a \vee b$ to illustrate how consistency maintenance is carried out for W-consistency to all the possible situations. Table 3.1 shows the exhaustive list of permutations

for the proposition-value for a, b and $a \vee b$ with the remarks column indicating the main consistency maintenance procedures if it is inconsistent. Those cases marked with # indicate that they are not the possible outcomes for NLBN. Consistency maintenance for a pure disjunctive logical expression (not necessarily restricted to a two input disjunction) are carried out under the following general categories corresponding to the cases in Table 3.1.

1. When the combined proposition value of the set of sub-expressions supports the logical expression, it is W-consistent. (Cases 1 to 4, 7 and 27)

2. When all of the proposition-values of the sub-expressions are (0,0), it is W-consistent because there is no weakly disagreeing sub-expressions and no influencing sub-expressions. (Case 5 and 14)

3. When a disjunction is believed and the negations of all its disjuncts are also believed, it is W-inconsistent as the combined input proposition value of (0,1) opposes the disjunction. To maintain W-consistency, all weakly disagreeing and influencing sub-expressions must be suppressed to (0,0) by a non-supporting logical suppression. Consistency check is carried out on the affected beliefs. (Case 9)

4. When the combined proposition value of the set of sub-expressions does not support the logical expression, and there are weakly disagreeing sub-expressions or influencing sub-expressions, it is W-inconsistent. To maintain consistency, all of the weakly disagreeing sub-expressions and

influencing sub-expressions are suppressed to (0,0). (Case 6, 8, 11-13, 15-18)

5. When the negation of a disjunctive expression is introduced into the belief state, the negation of each of its disjuncts are also believed at the same degree-of-belief value as the disjunction. For example, $\neg(a \vee b \vee c)$ is equivalent to $\neg a \wedge \neg b \wedge \neg c$. Therefore, from NLBN's computation, for the negation of the disjunction to be believed, the negation of all its disjuncts must also be believed. This makes all other combinations impossible and they need not be considered for consistency maintenance. (Cases 19 to 26)

The above can be extended to covered all possible consistency maintenance situations for a disjunctive expression.

Since the connective AND is logically symmetrical to OR in NLBN, consistency maintenance for conjunctive expressions can be carried out in the same ways. Table 3.2 shows the exhaustive list of permutations for the proposition-value for a, b and $a \wedge b$.

Consistency maintenance for a pure conjunctive logical expression (not necessarily restricted to a two input conjunction) are carried out under the following general categories corresponding to the cases in Table 3.2.

1. When the combined proposition value of the set of sub-expressions

Case	a b	$a \wedge b$		Remarks
1	(0,1) (0,1)	(0,1)	*	*: W-consistent (logically consistent).
2	(0,1) (0,0)	(0,1)	*	†: may be W-inconsistent—suppress all
3	(0,1) (1,0)	(0,1)	*	weakly disagreeing sub-expression(s) to
4	(0,0) (0,1)	(0,1)	*	(0,0).
5	(0,0) (0,0)	(0,1)	*	§: W-inconsistent—suppress all weakly
6	(0,0) (1,0)	(0,1)	†	disagreeing and interfering sub-express-
7	(1,0) (0,1)	(0,1)	*	ion(s) to (0,0).
8	(1,0) (0,0)	(0,1)	†	#: not applicable.
9	(1,0) (1,0)	(0,1)	§	
10	(0,1) (0,1)	(0,0)	§	
11	(0,1) (0,0)	(0,0)	§	
12	(0,1) (1,0)	(0,0)	§	
13	(0,0) (0,1)	(0,0)	§	
14	(0,0) (0,0)	(0,0)	*	
15	(0,0) (1,0)	(0,0)	†	
16	(1,0) (0,1)	(0,0)	§	
17	(1,0) (0,0)	(0,0)	†	
18	(1,0) (1,0)	(0,0)	§	
19	(0,1) (0,1)	(1,0)	#	
20	(1,1) (0,0)	(1,0)	#	
21	(0,1) (1,0)	(1,0)	#	
22	(0,0) (0,1)	(1,0)	#	
23	(0,0) (0,0)	(1,0)	#	
24	(0,0) (1,1)	(1,0)	#	
25	(1,0) (0,1)	(1,0)	#	
26	(1,0) (0,0)	(1,0)	#	
27	(1,0) (1,0)	(1,0)	*	

Table 3.2: All possible case for a two input AND relation $a \wedge b$

supports the logical expression, it is W-consistent. (Cases 1 to 4, 7 and 27)

2. When all of the proposition-values of the sub-expressions are (0,0), it is W-consistent because there is no weakly disagreeing sub-expressions and no influencing sub-expressions. (Case 5 and 14)

3. When the negation of a conjunction is believed and its conjuncts are also believed, it is W-inconsistent as the combined input proposition

value of (1,0) opposes the conjunction. To maintain W-consistency, all weakly disagreeing and influencing sub-expressions must be suppressed to (0,0) by a non-supporting logical suppression. Consistency check is carried out on the affected beliefs. (Case 9)

4. When the combined proposition value of the set of sub-expressions does not support the logical expression, and there are weakly disagreeing sub-expressions or influencing sub-expressions, it is W-inconsistent. To maintain consistency, all of the weakly disagreeing sub-expressions and influencing sub-expressions are suppressed to (0,0). (Case 6, 8, 11-13, 15-18)

5. When a conjunctive expression is introduced into the belief state, each of its conjuncts are also believed at the same degree-of-belief value as the conjunction. Therefore, from NLBN's computation, for the conjunction to be believed, all its conjuncts must also be believed. This makes all other combinations impossible and they need not be considered for consistency maintenance. (Cases 19 to 26)

As NLBN only accepts logical expression in CNF and only three layers of base nodes (the first layer contains all literals, the second layer contains all disjunctions and the third layer contains all conjunctions) and two layers of combinative-links (OR-links and AND-links) are necessary for representing all logical expressions.

3.5 The W-Consistency Reasoning Process

The goals of the *Consistency Reasoning Process* are to reason about inconsistencies in belief states, to resolve inconsistencies by making minimum adjustments to beliefs (via logical suppression(s) to the weakest relevant belief(s)), and to prevent unnecessary suppressions of beliefs. It is carried out whenever there is a belief change on the belief state. This reasoning process is a higher level procedure consists of two sub-processes: consistency check and consistency maintenance. Given an initial belief state, it starts with a consistency check to determine which are the inconsistent beliefs. Inconsistent beliefs then go through consistency maintenance process. After that it returns consistency check and it then loops until the belief state is consistent, or all inconsistent beliefs are exhaustively suppressed.

Given that a NLBN is a representation of a finite set of beliefs and the consistency reasoning process is built on a procedure that more and more logical suppressions are introduced from the logical expressions to its sub-expressions until it is consistent, when consistency reasoning progressed, more and more beliefs will be suppressed to an uncertain state. As the number of logical suppressions which can be carried out by each logical expression is limited by the number of sub-expressions in them, there will be a time when no more suppression is possible, then **W-Consistency reasoning process will terminate finitely and since exhaustively suppressed beliefs are con-**

sistent, it will yield a unique consistent belief state. Detailed proof is given in the next section.

3.6 Proof of Consistency Reasoning Process Terminates Finitely and Consistent

In this section, we give a proof which show that the consistency reasoning process terminates finitely and always resulting in a consistent belief state, we need the following definitions and lemmas.

Definition 1 (W-counter Suppression) A logical suppression from a logical expression Ψ, $\alpha \xrightarrow{\Psi} (\beta, \deg[\alpha])$, is W-counter suppressed when it has been suppressed by another stronger logical suppression, with degree-of-belief value greater than Ψ, to a belief inconsistent with Ψ. That is, there exists at least another logical suppression $\alpha \xrightarrow{\Theta} (\gamma, \deg[\gamma])$ where $\gamma \neq \beta$, $\deg[\gamma] > \deg[\Psi]$ and γ is inconsistent with Ψ.

Lemma 2 A fully suppressed logical expression is consistent.

Proof: A fully suppressed logical expression means that all its sub-expressions are suppressed to a contradictory state as only non-supporting logical suppressions are used. For a fully suppressed logical expression Ψ with n sub-expression, if there are less than or equal to $n-1$ numbers of W-counter

suppressions blocking the effects of the non-supporting logical suppressions from Ψ, the last non-supporting logical suppression still makes Ψ consistent because there are still no weakly disagreeing or influencing sub-expressions. On the other hand, if after full suppressions, all of them are W-counter suppressed by other stronger disagreeing beliefs, the negation of Ψ will be believed in the belief state. It is also consistent.

□

Lemma 3 An exhaustively suppressed logical expression is consistent.

Proof: If a logical expression Ψ with n sub-expressions is fully suppressed, from Lemma 2, it is consistent. Ψ is not being fully suppressed when it has k logical suppressions ($k < n$) and $n - k$ sub-expressions are not logically suppressed by Ψ. For W-consistency reasoning, when after exhaustively suppressing the weaker relevant opposing or influencing k sub-expressions from Ψ, all of them are W-counter suppressed by other stronger beliefs to some opposing beliefs stronger than Ψ and no more suppression is possible, this means that all sub-expressions of Ψ are opposing it and have higher degree-of-belief values than Ψ. In this case, the negation of Ψ will be believed and it is logically consistent.

□

Lemma 4 The maximum number of logical suppressions can be imposed on a belief state is the sum of all the number of sub-expressions in each logical

expression.

Proof: First, the maximum limit of logical suppressions can be imposed by each logical expression equals to the number of its sub-expressions. If maximum logical suppression is imposed for all logical expressions in a belief state (full suppression), then the total number of logical suppressions is the sum of the number of sub-expressions from all logical expressions.

□

Lemma 5 The W-consistency reasoning process terminates at a consistent belief state.

Proof: From the procedures of consistency reasoning process, there could be two modes of termination:

1. It terminates normally. That is, it terminates when there is no more inconsistent beliefs in the belief state.

2. It terminates when inconsistent beliefs are exhaustively suppressed. Following from Lemma 3 that exhaustively suppressed beliefs are consistent, the belief state that consists of consistent beliefs and the exhaustively suppressed beliefs are consistent.

□

Theorem 6 (Termination of consistency reasoning process) A consistency reasoning process terminates finitely in a consistent belief state.

Proof: From the basic definition, a Neural-Logic Belief Network is a representation of a finite set of beliefs and from Lemma 4, the consistency reaoning process terminates finitely with finite total number of possible suppressions. Following from this and Lemma 5, a consistency reasoning process will terminate finitely in a consistent state.

□

Chapter 4

Implementation

4.1 Introduction

XHOPES is a graphical user interface of a general knowledge-based expert system shell which aims to provide a user-friendly environment for the users to manipulate and access knowledge constructed in NLBN language.

In NLBN language, knowledge is constructed in terms of a graph, with nodes and links as basic elements. The graphical interface accepts the graph representation of the knowledge in NLBN language format and eliminates the need for the users (knowledge engineers) to transform the knowledge into statements, thus improves accuracy and reduces time. Also, queries to the system is made visible through the graphical interface, and the users can have a better understanding of the operations of the system. One of the snapshot of XHOPES interface is shown in Figure 4.1.

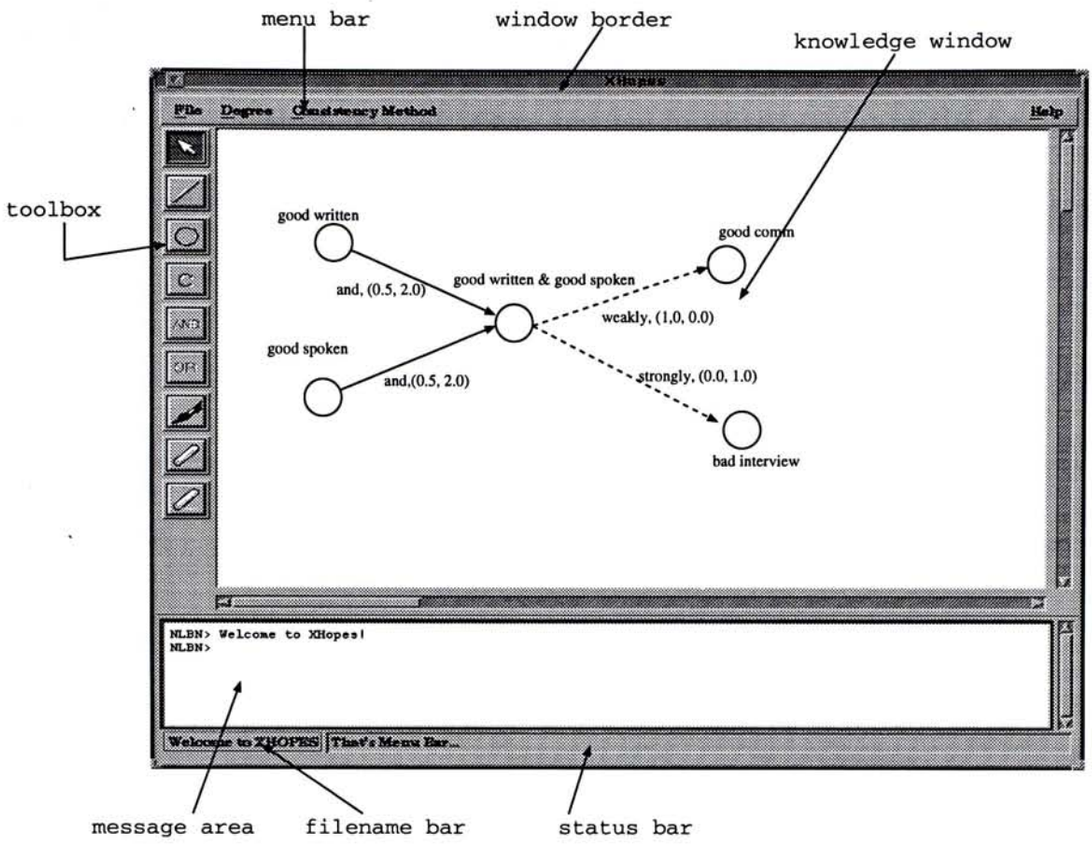


Figure 4.1: A snapshot of XHOPES

As many conventional knowledge systems employed text mode communication as interfaces, it is also desired to provide a text interface to promote compatibility for users of other systems. In addition, the text interface can act as a verification channel for the graphical interface and display the semantics that the graphical interface represents.

We use C++ to build the inference engine and interface of XHOPES is mainly implemented by OSF/MOTIF Release 1.2, which is run under the SUN X-window system. It is developed in two phases. In the first phase,

the basic inference engine (without consistency reasoning) and the graphical interface is built. In the second phase, we incorporate the consistency reasoning into the system so that it can resolve inconsistencies in the knowledge base and enhances the interface so that users can use it more efficiently.

For basic structure of inference engine on update operations and how the system interacts with the users, refer to the report of the first phase [2]. In this Chapter, we mainly focus on new features about consistency reasoning added in the second phase and see how the consistency reasoning process is implemented in the system.

4.2 New Features in Phase Two

4.2.1 Consistency Reasoning Function

In phase two, we mainly introduce the consistency reasoning process into the system. The consistencies we added in are G-Consistency, W-Consistency and S-Consistency [17] (which is a stronger version of consistency than both G- and W-Consistency). There are two ways we can assign the consistency type to each node in XHOPES. First, we can assign a particular consistency globally to all nodes. Second, we can assign a consistency to a specific node which may use a different consistency type from other nodes.

In the first way, we can choose the "Consistency Method" of menu bar and "Default Consistency Method" and then assign one consistency type to all nodes before any nodes are defined as shown in Figure 4.2. After specified, all nodes generated will have that consistency type as default.

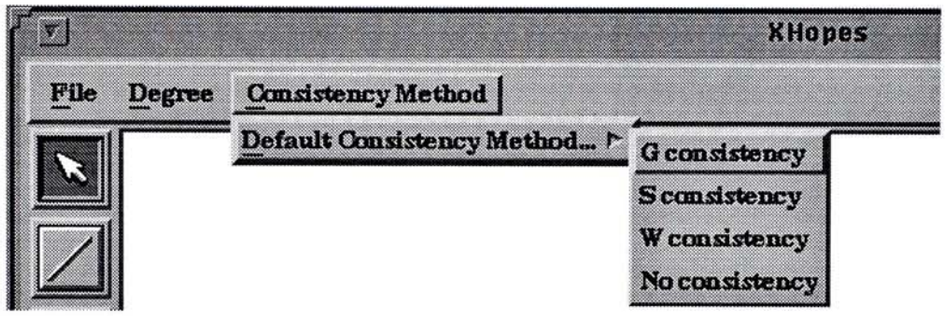


Figure 4.2: Choose Default Consistency Type

The second way is used when a different consistency type is applied to a particular node. The method is using the pop-up menu for that node, choose "Consistency Reasoning ..." and then pick up a consistency type as shown in Figure 4.3.

4.2.2 Knowledge File

The knowledge file is used to capture and store knowledge (or facts) of the real world as the knowledge base of XHOPES. Any nodes and links in the graphical window will be translated into knowledge file for storage so that

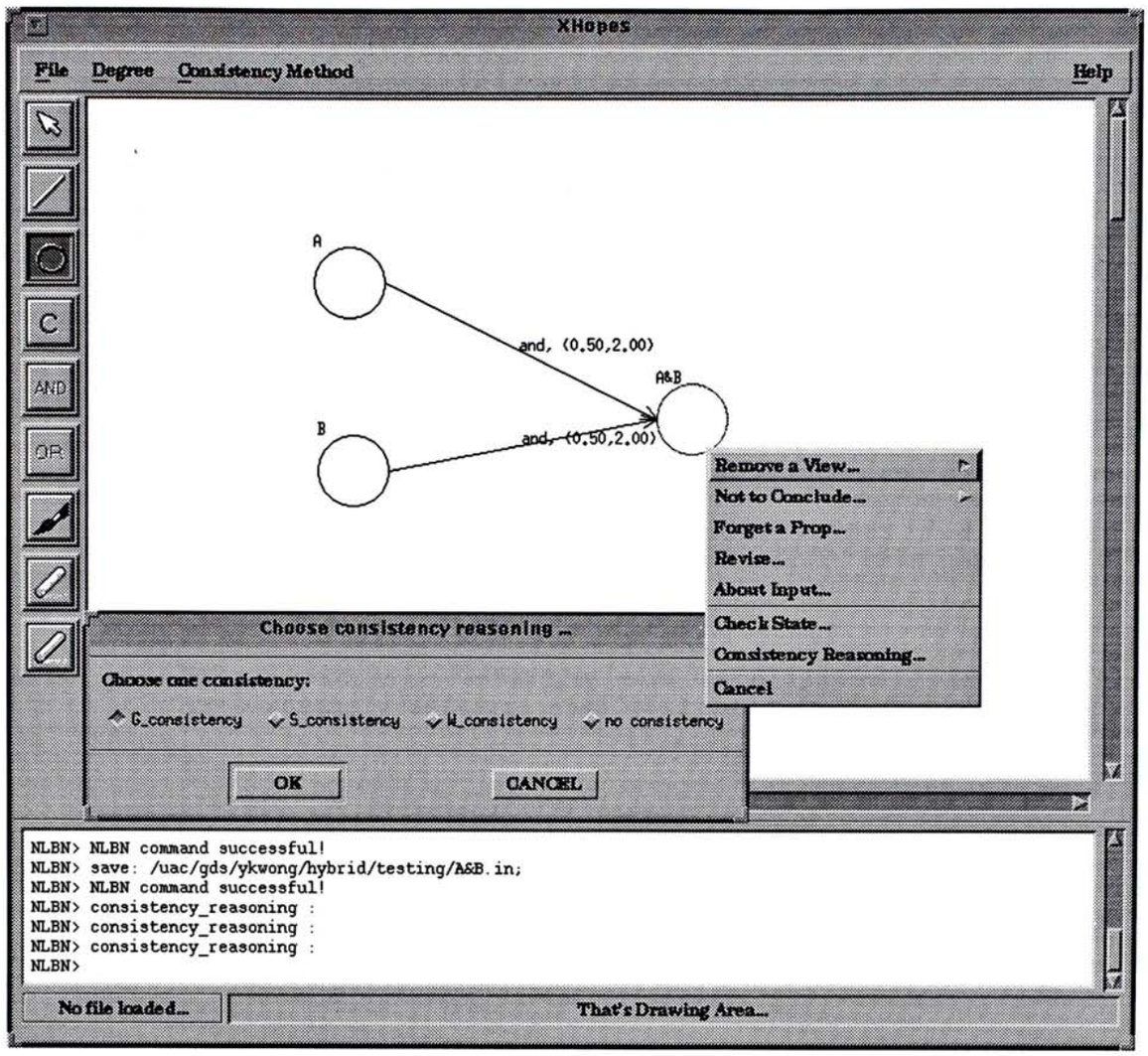


Figure 4.3: Choose Particular Consistency Type for a Node

XHOPES can retrieve any knowledge at any time. The knowledge file should be in a format that can be understood by human and interpreted by the parser of the XHOPES. The file should also allow editing through ordinary text editors as the knowledge base is likely to have future modifications.

Figure 4.4 is a simple knowledge file written for XHOPES. It describes the knowledge related in the determinant factors in the job hunting process.

```
begin{

    define degree :=
           definitely(1.00)>strongly(0.79)>mostly(0.56)>normally(0.27)>
    probably(0.24);

    define good_job := "can find a good job";
    define good_comm := "good communication skills";
    define good_spoken := "good oral communication";
    define good_result := "good result in honours";
    define good_written := "good written skills";

    rule:
           it's normally believed that if good_written and good_spoken
    then good_comm;
           it;s strongly believed that if good_comm then good_job;
           it;s strongly believed that if good_result then good_job;
    end;

    belief:
           it's normally believed that not (good_spoken);
           it's strongly believed that good_written;
    end;

}
```

Figure 4.4: Sample knowledge file

The knowledge file is divided into four main parts. The users will find the *degree* part at the top. This is because the degrees defined here will be used later in the *rule* part and *belief* part. The degrees are used to compare the relative magnitude of rules and beliefs in the aforesaid parts respectively.

The next part is the *variable* part. All the propositions that will be used in the knowledge file must be defined in this part. It enhances the work of the parser of XHOPES by enforcing the users to give labels to the propositions. For example, *good_written* is a label for the proposition "good written skills" that can be found in the above example knowledge file.

The *rule* part is used for capturing all IF-THEN rule in the knowledge base. For example, we have *it's normally believed that if good_written and good_spoken then good_comm* in the knowledge file. Here, *normally* is a degree defined in the *degree* section, and *good_written*, *good_spoken* and *good_comm* are labels defined in the *variable* section. The condition and/or conclusion phrases of the IF-THEN rule can be composed by more than one label using logical AND/OR as connectives.

The *belief* part represents external input to the system. These inputs from the environment give excitations to the system. A belief can be composed by more than one label using logical AND/OR as connectives. Negation can also be used in composing beliefs by a leading *not* followed by the belief in parentheses '(' and ')'.

There are some minor differences in the knowledge file structure written by XHOPES. This format will be used whenever the system saves a newly created knowledge or when the system updates the knowledge modified by the users. Figure 4.5 is a knowledge file, about determinant factors in the job hunting process, saved by XHOPES.

The pair of numbers following each definition of the variable is the geometric location of corresponding nodes in the knowledge window of XHOPES.

```
begin{
   define degree :=
          definitely(1.00)>strongly(0.79)>mostly(0.56)>normally(0.27)>
   probably(0.24);

   define good_job := "can find a good job"; (682, 250);
   define good_comm := "good communication skills"; (478, 118);
   define good_spoken := "good oral communication"; (99, 279);
   define good_result := "good result in honours"; (402, 411);
   define good_written := "good written skills"; (112, 143);

   rule:
          good_written and good_spoken;
          it's normally believed that if good_written and good_spoken
   then good_comm;
          it;s strongly believed that if good_comm then good_job;
          it;s strongly believed that if good_result then good_job;
   end;

   belief:
          it's normally believed that not (good_spoken);
          it's strongly believed that good_written;
   end;

   state:
          good_spoken&good_written := negative, normally, yes, yes, w consistency;
          good_result := uncertain, uncertain, yes, yes, w consistency;
          good_spoken := negative, normally, yes, yes, g consistency;
          good_comm := uncertain, uncertain, yes, yes, g consistency;
          good_job := uncertain, uncertain, yes, yes, g consistency;
   end;

}
```

Figure 4.5: Sample knowledge file

For the *rule* part, base nodes created by the system itself will be specified in advanced. As seen from the knowledge file e.g. *good_written and good_spoken*, stands for the base node created by the system resulted from the combinative links (logical AND) from the node *good_spoken* and *good_written*. This node will be assigned with a label named *good_spoken&good_written*, and will be used by the system in subsequent rule definitions. The new part added is the *state* part which is used to record all states of nodes when they are saved. The format is as followings:

label := proposition-value, degree, reliable, consistency checked, consistency

type;

where *reliable* can be {yes, no}. It is used to record whether it is reliable to use after some changes on the network. *Consistency checked* can be {yes, no}. It is used to record whether it has been checked for consistency or not.

4.3 Inference Engine for Consistency Reasoning

The main responsibility of the inference engine is computing state of the whole NLBN network everytime the user requests information. It also performs consistency reasoning if required. It is implemented using C++. In [2], the detail information of class structure and how update operations implemented are described. In this section, we will mainly go through the process of how it computes the state of nodes and how it performs consistency reasoning.

A node i is directly related to another node j if there is a link (combinative link or rule link) connecting these two nodes. A node i is indirectly related to node j if node i can be traced backward or forward through the links and somehow reaches node j. Given a node in NLBN, we can get a set of nodes which is related to it. Therefore, we can always extract a portion of

the network S_p from S which contains the maximum number of nodes that are related and we call it *maximal related sub-network.* When there is an update operation on any node, say i. XHOPES will turn *reliable* and *consistency check* flags of all nodes in maximal related sub-network of i to the state of *'No'*. At this stage, XHOPES just performs the update operation on node i and does not feed forward the changed node value to other nodes. When the users explicitly click the "*evaluate state*" function on the pop up menu of a node, j say. If the *reliable* flag of j is *'Yes'*, XHOPES just output the result. If the flag is *'No'*, XHOPES uses the inference engine to calculate the node values of required nodes. The algorithm is shown in Figure 4.6.

re-evaluate() function is used to move forward or backward in a network and enhances the calculation of the states of nodes. It is called recursively until it reaches nodes that are reliable to feed the node values forward. When all the sub-expressions and conditions of rules of node i are reliable, we can safely calculate the state of node i. *calculate_state()* function is used to calculate the state of the node, say i, in the assumption that all previous nodes affecting i is reliable. After calculation of state of the node, the *consistency_reasoning_process()* is called to check the consistency. In this function, the process of consistency maintenance continues as long as the node is inconsistent.

```
re-evaluate (node i)
{
   if(reliable_of_i == YES and consistency_check_of_i == YES)
   {
      /* do nothing and return */
   }

   else
   {
      collect all sub-expressions, j say, of i;
      for(each sub-expressions j)
      {
          re_evaluate (node j);
      }

      collect all condition nodes, k say, which have rule links and
         use node i as conclusion node;
      for(each condition k)
      {
          re_evaluate (node k);
      }

      calculate_state (node i);
      consistency_reasoning_process (node i);
      change reliable and consistency_check flag to YES;
    }
}


consistency_reasoning_process (node i)
{
   while(consistency_check (node i))
   {
      consistency_maintenance (node i);
   }
}
```

Figure 4.6: Algorithm of Inference Engine of XHOPES

4.4 Examples of using XHOPES

In this section, a number of benchmark problems for nonmonotonic reasoning are extracted from [13] for illustrating how XHOPES can be used to formalize these problems. Note that the representation in each problem is only one of the solutions and different representations can be used for the same problem in NLBN. The description of each problem consists of a list of assumptions, followed by one or more conclusions and then the representation in NLBN knowledge file.

Example 1: Default Reasoning with Several Defaults

Assumptions:

Blocks A and B are heavy.

Heavy blocks are normally located on the table.

Heavy blocks are normally red.

A is not on the table.

B is not red.

Conclusions:

B is on the table.

A is red.

NLBN Representation:

```
begin
{
define degree := true(1.00) > definitely(0.89) > strongly(0.73) > normally(0.52)
> usually(0.32) > weakly(0.14);
define A_red := "A is red.";
define B_red := "B is red.";
define A_on_table := "A is on the table.";
```

define B_on_table := "B is on the table.";

define A_heavy := "Block A is heavy.";

define B_heavy := "Block B is heavy.";

belief:

it's normally believed that A_heavy;

it's normally believed that B_heavy;

it's definitely believed that not (B_red);

it's definitely believed that not (A_no_table);

end;

rule:

it's normally believed that if B_heavy then B_red;

it's normally believed that if B_heavy then B_on_table;

it's normally believed that if A_heavy then A_red;

it's normally believed that if A_heavy then A_on_table;

end;

}

Results of NLBN:

A is red. (normally)

B is not red. (definitely)

A is not on the table. (definitely)

B is on the table. (normally)

The result is the same as the conclusion made in [13].

Example 2: Reasoning with Incomplete Information

Assumptions:

At least one of the blocks A, B is on the table.

Conclusion:

It is not known whether A is on the table.

It is not known whether B is on the table.

NLBN Representation:

```
begin
{
define degree := true(1.00) > definitely(0.89) > strongly(0.73) > normally(0.52)
> usually(0.32) > weakly(0.14);
define A_on_table := "A is on the table.";
define B_on_table := "B is on the table.";
rule:
A_on_table or B_on_table;
end;
belief:
```

it's strongly believed that A_on_table|B_on_table; end;

}

Results of NLBN:

A is on the table. (uncertain)

B is on the table. (uncertain)

A or B is on the table. (strongly)

The result is the same as [13] and only the disjunction is believed.

Example 3: Linear Inheritance

Assumptions:

Animals normally do not fly.

Birds are animals.

Birds normally fly.

Ostriches are birds.

Ostriches normally do not fly.

Conclusions:

Animals other than birds do not fly.

Birds other than ostriches fly.

Ostriches do not fly.

NLBN Representation:

begin

{

define degree := true(1.00) > definitely(0.89) > strongly(0.73) > normally(0.52)

> usually(0.32) > weakly(0.14);

define Ostriches := "Ostriches";

define Birds : "Birds";

define Animals := "Animals";

define can_fly := "They can fly";

rule:

it's true that if Ostriches then Birds;

it's true that if Birds then Animals;

it's strongly believed that if Birds then can_fly;

it's normally believed that if Animals then not (can_fly);

end;

belief:

it's true that Ostriches;

it's strongly believed that not (can_fly);

end;

}

Results of NLBN:

Ostriches are birds; (true)

Ostriches are animals; (true)

Ostriches cannot fly; (definitely)

IN this example, we cannot conclude "*Animals other than birds do not fly*" and "*Birds other than ostriches fly*" because NLBN uses propositional logic, it is difficult to represent general rule with quantifiers in the knowledge base. Despite this drawback, we see that the inference made by NLBN is similar to the conclusions should by drawn in the benchmark problems.

In conclusion, we have successfully built a prototype of knowledge representation and reasoning system shell based on NLBN. It provides a graphical user interface for manipulation of knowledge in an efficient way. Although XHOPES lacks the ability of represent rules with quantifiers, it is flexible to model many kinds of specific common sense as shown in examples. New features could be added to enhance the functionality of the system in a later stage.

Chapter 5

Comparison between NLBN with W-Consistency and AGM Logic

In this chapter, we review several approaches for belief revision and resolving inconsistencies in knowledge base and make a comparison between NLBN with W-Consistency and these approaches. This review and comparison gives us a better understanding of different approaches for belief revision.

5.1 AGM Logic with Epistemic Entrenchment

5.1.1 Three Forms of Belief Change

AGM Logic [5] considers that a belief state is represented by a set of sentences from a language L which is based on first order logic and is closed under logical consequence. There are three main kinds of belief changes defined on a belief set K:

1. *Expansion*: A new sentence A is added to a belief system K and is denoted K_A^+.

2. *Contraction*: A sentence A is retracted without adding any new facts and is denoted K_A^-.

3. *Revision*: A new sentence A that is inconsistent with a belief system K is added and some of the old sentences in K must be retracted in order that the final belief system is consistent and is denoted K_A^*.

AGM Logic has formulated postulates for these three belief revision operations:

Postulates for expansions are:

(K+1) For any sentence A and belief set K, K_A^+ is a belief set

(K+2) A belongs to K_A^+

(K+3) $K \subseteq K_A^+$

(K+4) If $A \in K$, then $K_A^+ = K$

(K+5) If $K \subseteq H$, then $K_A^+ \subseteq H_A^+$

(K+6) For all belief sets K and all sentences A, K_A^+ is the smallest belief set that satisfies $(K+1) - (K+5)$.

Postulates for revisions are:

(K*1) For any sentence A and belief set K, K_A^* is a belief set

(K*2) A belongs to K_A^*

(K*3) $K_A^* \subseteq K_A^+$

(K*4) If $\neg A \notin K$, then $K_A^+ \subseteq K_A^*$

(K*5) $K_A^* = K_\perp$ if and only if $\vdash \neg A$ (where $K_\perp$ is the absurd belief set)

(K*6) If $\vdash A \leftrightarrow B$, then $K_A^* = K_B^*$

(K*7) $K_{A \wedge B}^* \subseteq (K_A^*)_B^+$

(K*8) If $\neg B \notin K_A^*$, then $(K_A^*)_B^+ \subseteq K_{A \wedge B}^*$

Postulates for contractions are:

(K-1) For any sentence A and any belief set K, K_A^- is a belief set

(K-2) $K_A^- \subseteq K$

(K-3) If $A \notin K$, then $K_A^- = K$

(K-4) If not $\vdash A$, then $A \notin K_A^-$

(K-5) If $A \in K$, then $K \subseteq (K_A^-)_A^+$

(K-6) If $\vdash A \leftrightarrow B$, then $K_A^- = K_B^-$

(K-7) $K_A^- \cap K_B^- \subseteq K_{A \wedge B}^-$

(K-8) If $A \notin K_{A \wedge B}^-$, then $K_{A \wedge B}^- \subseteq K_A^-$

Expansions can simply be defined as the logical closure of K together with A:

$$K_A^+ = \{B : K \cup \{A\} \vdash B\}$$

It is not possible to give a similar explicit definition of revisions and contractions in logical and set-theoretical notions only. Contraction and revision are interrelated through Levi's and Harper's identity:

$$K_A^* = (K_{(\neg A)}^-)_A^+$$

$$K_A^- = K \cap (K_{(\neg A)}^*)$$

Therefore, revision functions can be described through the associated contraction function and conversely.

5.1.2 Epistemic Entrenchment

The rationality postulates of AGM Logic give the guideline to construct the belief revision operations. They tell us what should the belief revision operations do when a belief state changes. Based on these postulates, there are several ways to construct the operations based on these postulates [5].

One of the well known approach is using *epistemic entrenchment.* The basic idea of this approach is that the sentences that are accepted in a given belief set K have different degrees of *epistemic entrenchment* – not all sentences that are believed to be true are of equal value for planning or problem-solving purposes, but certain pieces of our knowledge and beliefs about the world are more important than others when different situations occur. The guiding area of constructing a contraction or revision function is that when a belief set K is revised or contracted, the sentences in K that are given up are those having the lowest degrees of epistemic entrenchment. If A and B are sentences in L, the notation $A \leq B$ will be used as a shorthand for "B is at least as epistemic ally entrenched as A". [5] has the following postulates for epistemic entrenchment:

(EE1) If $A \leq B$ and $B \leq C$, then $A \leq C$

(EE2) If $A \vdash B$, then $A \leq B$

(EE3) For and A and B, $A \leq A \wedge B$ or $B \leq A \wedge B$

(EE4) When $K \neq K_{\perp}$, $A \notin K$ iff $A \leq B$, for all A

(EE5) If $B \leq A$ for all B, then $\vdash A$

The relationships between the ordering of epistemic entrenchment and the contraction and revision functions are given by (C$\leq$) and (C$-$). (C$\leq$) determines an ordering of epistemic entrenchment assuming a contraction function and a belief set as given, and (C$-$) determines a contraction function assuming an ordering of epistemic entrenchment and a belief set as given.

(C$\leq$) $A \leq B$ if and only if $A \notin K^{-}_{A \wedge B}$ or $\vdash A \wedge B$

(C$-$) $B \in K^{-}_{A}$ if and only if $B \in K$ and either $A < A \vee B$ or $\vdash A$

Therefore, the problem of constructing appropriate contraction and revision functions can be reduced to the problem of forming an appropriate ordering of epistemic entrenchment given a belief set K.

5.2 Network Update Operators in NLBN vs. Belief Changes in AGM

NLBN uses six network update operators (*Add*, *Remove*, *Forget*, *Revise*, *Update* and *Not-Conclude*) [18] to change the belief state. Performing an update

operation for π with respect to a belief a on a NLBN with a belief state K [1] is denoted as $\pi(a, K)$ and the resulting belief state is denoted as K_a^{π}. In the following discussion, a belief, say a, is associated with a proposition-value and a degree-of-belief value. Belief a and belief b may be the same proposition with different proposition-values and different degree-of-belief values.

A new belief is incorporated into the network via an *Add* operation. This operation corresponds to *Expansion* in AGM Logic. Obviously, When a belief a is added into the K, K_a^{Add} is still a belief set in NLBN and if a is already in K, $K_a^{Add} = K$. Moreover, when a is added into the belief state K, only those required to be in K_a^{Add} will be present. Other beliefs that are not affected by a are not introduced into K_a^{Add}. As a result, the *Add* operation of NLBN satisfies (K+1), (K+4) and (K+6) of AGM Logic.

On the other hand, the added belief a may not in K if deg[a] is smaller than the degree-of-belief value of the same proposition originally present in K. This violates (K+2) that A belongs to K_a^{Add}. *Add* also violates (K+3). For example, we have an inital belief state K which contains only one proposition : $\{a : ((1,0), weakly)\}$. If we add a belief a: ((0,1), strongly) into K, the new belief state K_a^{Add} becomes $\{a : ((0,1), strongly)\}$ and K is not a sub-set of K_a^{Add}. Moreover, (K+5) is not satisfied by *Add*. Given two belief

[1] $K = S$ and we use K here to conform with the notation of AGM Logic. Moreover, *belief state* and *belief set* are interchangeable

sets $K_1 = \{a : ((0,0),0)\}$ and $K_2 = \{a : ((1,0), strongly)\}$, $K_1 \subseteq K_2$. When we add a: ((0,1), weakly) into K_1 and K_2, $(K_1)_a^{Add} = \{a : ((0,1), weakly)\}$ and $(K_2)_a^{Add} = \{a : ((1,0), strongly)\}$ and $(K_1)_a^{Add} \not\subseteq (K_2)_a^{Add}$.

Remove operation can remove an input value or a particular view (positive or negative) of a proposition. If we write $Remove(a, K)$, we assume that it only retracts all necessary beliefs such that the positive view of a is removed. For example, a proposition a may have two inputs: ((0,1), weakly) and ((1,0), strongly). Current state of a is ((1,0), strongly). $Remove(a, K)$ only removes input value ((1,0), strongly). Therefore the new state of a is ((0,1), weakly). It corresponds to the *Contraction* in AGM Logic. Obviously, a belief state after a *Remove* operation is still a belief state in NLBN (postulate (K-1)). *Remove* operation can always succeed to retract the belief so that it does not appear in the new belief state (postulate (K-4)) and it does not alter the belief state when the belief being retracted is not present in the initial state (postulate (K-3)).

In general, we do not expect that the contracted belief state after a *Remove* operation is a subset of the original state, that is, $K_a^{Remove} \not\subseteq K$. This can be illustrated using one example as shown in Fig. 5.1. Remember that we have a TAO: $0 < weakly < probably < normally < strongly < true$ used in the example.

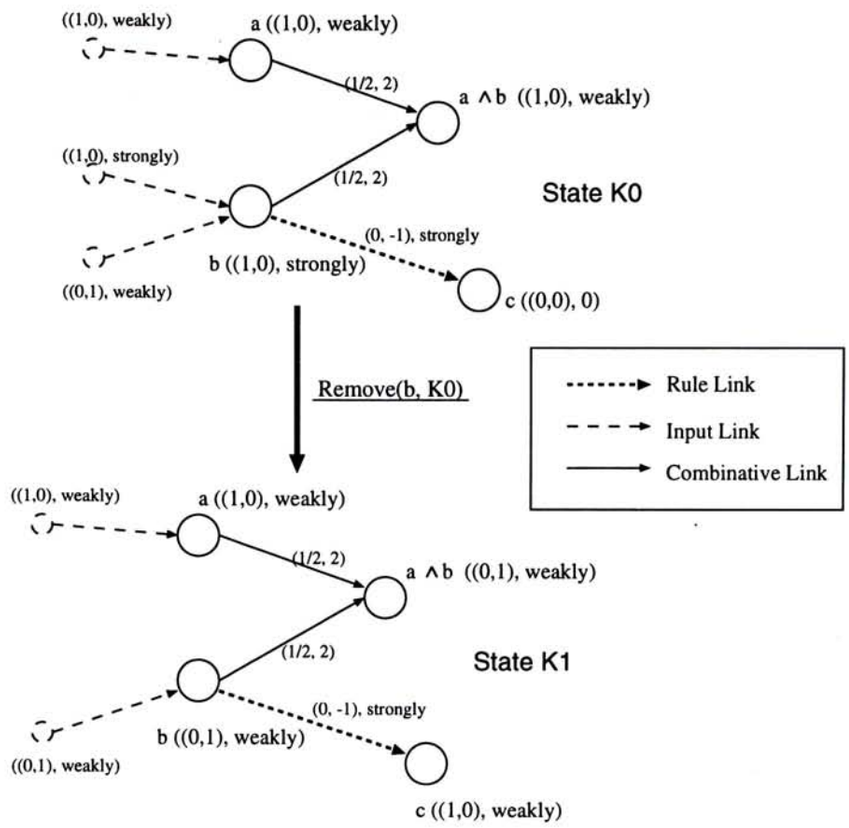


Figure 5.1: Example shows that *Remove* violates the postulate (K-2)

There are two inputs to b in K_0: $((1,0),\ strongly)$ and $((0,1),\ weakly)$ and the node value of b is the stronger input: $((1,0))$. If we retract positive view of b, the node value of B becomes $((0,1),\ weakly)$. It makes us believe c by the rule "IF not b THEN c". Therefore, $(K_0)_b^{Remove}$ has a belief that K_0 does not and it violates postulate (K-2).

Moreover, *Remove* operation does not follow the principle of recovery (postulate (K-5)) as shown in Fig 5.2. The initial state K_0 contains three propositions: a, b and $a \wedge b$. The initial belief state K_0 is shown in Table 5.1.

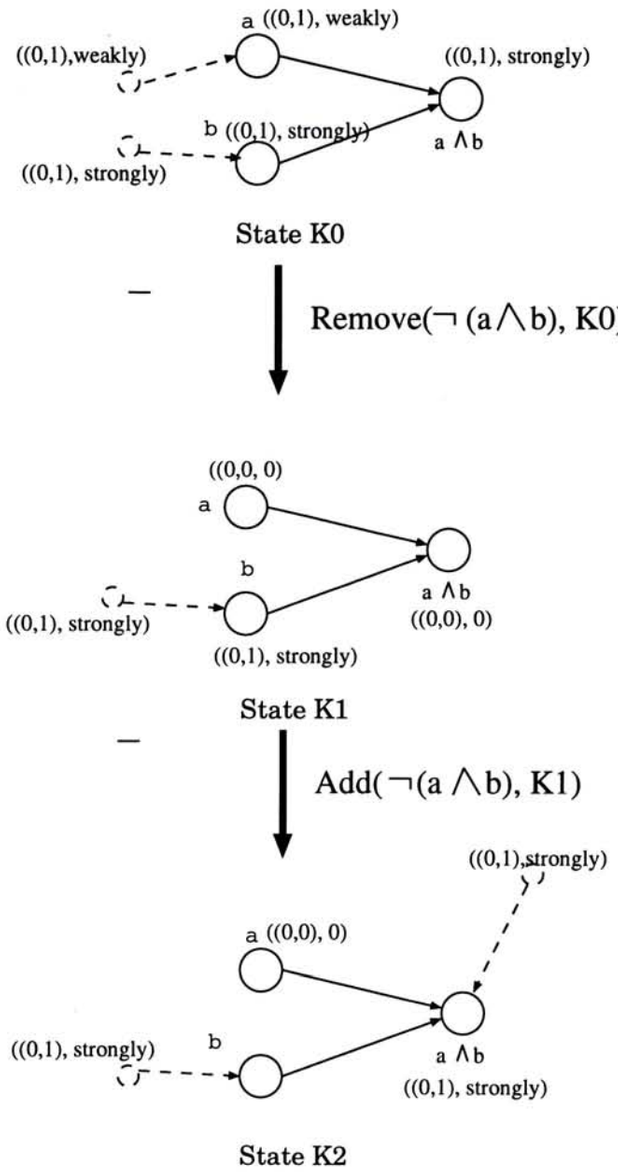


Figure 5.2: Example shows that *Remove* violates the principle of recovery

After $Remove(\neg(a \wedge b), K_0)$ is operated on K_0, Belief a: ((0,1), weakly) must be retracted as it is a weaker one that contributes support to $\neg(a \wedge b)$. Later, if we reverse the operation and re-insert $\neg(a \wedge b)$, that is, performing $Add(\neg(a \wedge b), K_1)$. Belief a: ((0,1), weakly) does not re-inserted. The propositions in K_2 are shown in Table 5.2.

Proposition	State
a	$((0,1)$, weakly$)$
b	$((0,1)$, strongly$)$
$a \wedge b$	$((0,1)$, strongly$)$

Table 5.1: Node Values of K_0: example shows *Remove* vioate (K-5)

Proposition	State
a	$((0,0)$, $0)$
b	$((0,1)$, strongly$)$
$a \wedge b$	$((0,1)$, strongly$)$

Table 5.2: Node Values of K_2: example shows *Remove* violate (K-5)

We see that the belief a: $((0,1)$, weakly$)$ is lost during these operations. Therefore, adding back a proposition which is removed previously does not restore the original belief state in NLBN.

Remove does not also follow the postulate (K-7). For example, given an initial belief state K_0 as shown in Table 5.3 where $a \wedge b$ has a direct input of $((0,1)$, strongly$)$.

$K_a^{Remove} = \{a{:}((0,0),\ 0),\ b{:}((1,0),$ normally$),\ a \wedge b{:}((0,1),$ strongly$)\ \}$

$K_b^{Remove} = \{a{:}((1,0),$ weakly$),\ b{:}((0,0),\ 0),\ a \wedge b{:}((0,1),$ strongly$)\ \}$

$K_a^{Remove} \bigcap K_b^{Remove} = \{a{:}((0,0),\ 0),\ b{:}((0,0),\ 0),\ a \wedge b{:}((0,1),$ strongly$)\ \}$

Proposition	State
a	((1, 0), weakly)
b	((1, 0), normally)
$a \wedge b$	((0, 1), strongly)

Table 5.3: Node Values of K_0: example shows *Remove* violate (K-7)

However, since the positive aspect of $a \wedge b$ is not in K_0, there will be no change to the original belief state. $K_{a\wedge b}^{Remove} = \{a$:((1, 0), weakly), b:((1, 0), normally), $a \wedge b$:((0, 1), normally)$\}$ and therefore, $K_a^{Remove} \wedge K_b^{Remove} \not\subseteq K_{a\wedge b}^{Remove}$.

When we compare *Remove* with postulate (K-8), we assume that $a \notin K_{a\wedge b}^{Remove}$ means that deg[a] $\leq$ deg[b] and in order to remove $a \wedge b$, the weaker belief a has to be removed. We can easily show that *Remove* does not follow (K-8). It is because $K_{a\wedge b}^{Remove}$ removes beliefs a and b for a limiting case that deg[a] = deg[b] and $\neg a$ and $\neg b$ may appear in the belief state. However, K_a^{Remove} does remove belief b and $\neg b$ does not appear. Therefore, $K_{a\wedge b}^{Remove} \not\subseteq K_a^-$.

Revise operation removes all current views to a belief and assert the new belief to it. It is the same as *Forget* a belief and *Add* a new belief to the belief

state. This operation is corresponds to *Revision* of AGM Logic. It is not difficult to see that *Revise* does not follow most of the postulates of revision of AGM Logic except (K*1) and (K*2) and (K*8). For (K*8), at the right hand side, revise $a \wedge b$ with deg$[a \wedge b]$ must make the belief into the belief state by removing all inputs from it and those inputs from sub-expressions that previously contribute a different belief to $a \wedge b$. Therefore, states of a and b must change to positive view after revision (because by adding $a \wedge b$). At the left hand side, revising a with deg$[a]$ has the same effect as described on the right hand side. Since $\neg b$ does not in K_a^{Revise}, the state of b must be (1,0) or (0,0), therefore, $(K_a^{Revise})_b^{Add}$ must contain positive view of b and finally a and b supports $a \wedge b$. The final result is that both sides contain positive views of a, b, and $a \wedge b$.

(K*5) and (K*6) cannot apply to *Revise* since NLBN do not have "$K_\perp$" and "$\vdash a \leftrightarrow b$". When we look at (K*3), since a may not be in K_a^{Add} while it must be in K_a^{Revise}, therefore, $K_a^{Revise} \not\subseteq K_a^{Add}$. For (K+4), $K_a^{Add} \not\subseteq K_a^{Revise}$ in general because *Add* a logical expression, $a \vee b$ say, does not have any effect on its sub-expressions when it is consistent. However, *Revise* $a \vee b$ may remove a particular view of its sub-expressions if that view supports the logical expression before *Add* $a \vee b$ in. To show that *Revise* does not follow (K*7), let us look at an initial belief state K_0 in Table 5.4.

If we want to revise $a \wedge b$ to $((1,0)$, weakly), all inputs to the above expres-

Proposition	Inputs	State
a	$((1,0)$, strongly)	$((1,0)$, strongly)
b	$((1,0)$, strongly)	$((1,0)$, strongly)
	$((0,1)$, normally)	
$a \wedge b$	$((0,1)$, strongly)	$((1,0)$, strongly)

Table 5.4: Node Values of K_0: example shows *Remove* violate (K*7)

sions have to be removed before the new belief is added and $K_{a\wedge b}^{Revise}$ contains a, b and $a \wedge b$ with $((1,0)$, weakly). However, if we only revise a to $((1,0)$, weakly), the negative view of b: $((0,1)$, normally) does not removed, and adding b: $((1,0)$, weakly) to the belief state does not have effect on the state of b since *normally* $>$ *weakly*. Therefore, $(K_a^{Revise})_b^{Add}$ contains b: $(((0,1)$, normally). It violates (K*7).

W-Consistency reasoning process always makes necessary changes in the belief state to maintain W-Consistency after each operation. *Forget* operation of a NLBN removes all views to the belief and expel it from the belief network whenever possible. This can be accomplished by *Removing* both the positive view and the negative view of the belief. *Revise* operation removes all current views to a belief and assert the new belief to it. It is the same as *Forget* a belief and *Add* a new belief to the belief state.

There is not corresponding operation in AGM Logic for *Update* and *Not-Conclude* in NLBN. *Update* operation changes the degree-of-belief value of an input of a belief. *Not-Conclude* operation can temporarily suppress a belief and it is simply the same as *Adding* an input $((0,0), true)$ to the belief state.

We have compared the operations on NLBN with AGM Logic and see that NLBN's update operators do not follow some of the rationality postulates of AGM Logic. The main reason is that NLBN allows both positive and negative information and different degrees of credibility of the same proposition in the same belief state. Retracting one of beliefs of a proposition may result in believing another belief and its consequences. This characteristic is contributed by the underlying Logic used in the representation (Section 5.5) and ways to deal with inconsistent beliefs (Section 5.6) in NLBN. In the next section, we compare the *Epistemic Entrenchment* of AGM Logic and Degree-of-Belief values of NLBN.

5.3 Epistemic Entrenchment vs. Degree-of-Belief

In human reasoning, not all beliefs are accepted as equal value for planning or problem-solving purposes. Some knowledge may be more important than

others in some situations. AGM Logic uses *epistemic entrenchment* and NLBN uses *degree-of-belief* values to handle this kind of ordering and apply them to guide the revision of belief states.

If AGM Logic, when a knowledge system K is revised or contracted, the sentences in K that have the lowest epistemic entrenchment are given up. The postulates of epistemic entrenchment make sure that changes in a knowledge system is minimal. The revision and contraction functions constructed using the epistemic entrenchment orderings also satisfy $(K*1)$-$(K*8)$ and $(K-1)$-$(K-8)$ respectively, the details of which can be found in [5].

Although the above result can help to reduce the construction a belief revision operator to forming an ordering of epistemic entrenchment over the belief set, there is a problem that this approach cannot be solved. AGM Logic with epistemic entrenchment cannot handle iterated belief change [24]. Iterated belief change is a situation that after a belief change due to a new information x, that belief state must be revised by further new information y. Since an epistemic entrenchment is relative to a particular belief set, assume that the initial belief set is K and its corresponding epistemic entrenchment ordering $\leq_K$. When x comes in, K becomes to K_x^* according to $\leq_K$. Later, K_x^* is required to change because of new information y. However, we have no way to compute $(K_x^*)_y^*$ in the original AGM Logic because we do not have the

way to construct an epistemic entrenchment ordering $\leq_{K_x^*}$ and so as $\leq_{(K_x^*)_y^*}$.

The *degree-of-belief* value represents the belief certainty for a proposition in NLBN. Similar to AGM Logic, NLBN only handles the minimum number of weaker beliefs to restore the consistency of a belief state.

It is straight forward that the degree-of-belief value satisfies (EE1) and (EE4). It induces a total asymmetric order (TAO) which is transitive. The default node value in NLBN is (0,0) with the weakest degree (usually use "0" to denote the weakest degree) which means that the corresponding proposition is neither believed nor not believed. Moreover, by definition of computation functions and construction of conjunctive relations, the node value of a conjunctive expression Ψ with its sub-expressions α_i is the minimum of $deg[\alpha_i]$. Therefore, the node value of a conjunctive expression is at least equal to one of its sub-expressions and obeys (EE3).

All propositions are defeasible and no tautology is represented by a NLBN so that we do not have (EE5). NLBN uses IF-THEN rule to model some beliefs are justifications of other beliefs. A rule "IF a THEN b (certainty of the rule)", written as $a \mapsto b(\deg[a \mapsto b])$, means if we believe a (condition), then we derive b (conclusion) with respect to the rule degree-of-belief value. It does not require that $deg[condition] \leq deg[conclusion]$. (EE2) is not satisfied.

The persistence characteristic of NLBN ensures that the degree-of-belief values of nodes will not be changed if it is not affected by any network update operation, and it is not relative to a particular belief state. There is always a unique degree-of-belief values ordering between propositions for the revised belief state. Therefore, NLBN can easily handle iterated belief change.

5.4 Consistency Preservation

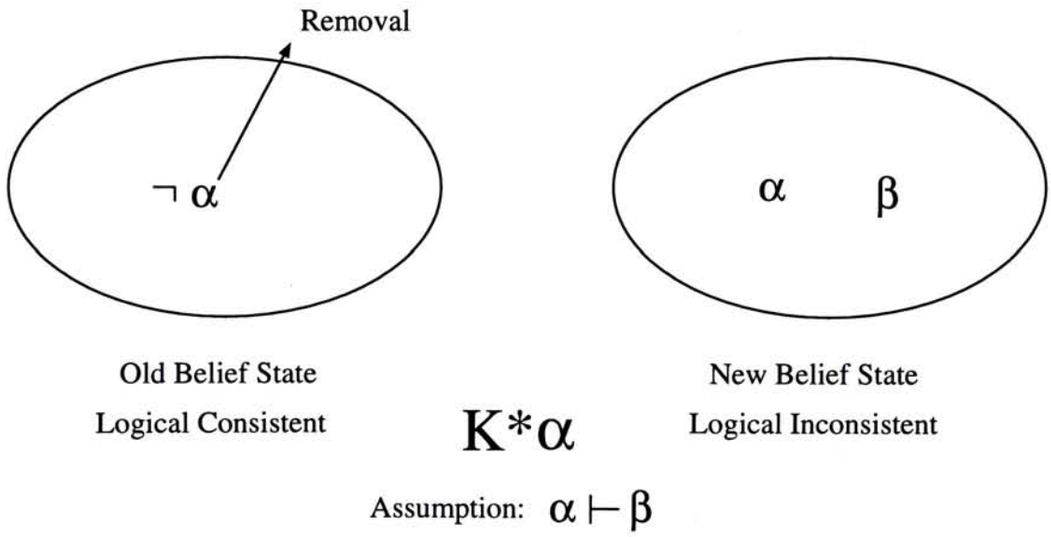


Figure 5.3: Revision of a Belief State

Consistency is an important consideration when we talk about the dynamic changes of a belief state. In AGM Logic, the postulate $(K * 5)$: $K_A^* = K_\perp$ if and only if $\vdash \neg A$ guarantees that the consistency of belief sets is preserved after a "consistent" revision. For example, in Figure 5.3, if

we want to revise K with α, $\neg\alpha$ must be taken out so that K^*_α is consistent. This is also true for contraction operation as contraction and revision can be interchangeable like Figure 5.4 in which α and β must be taken out.

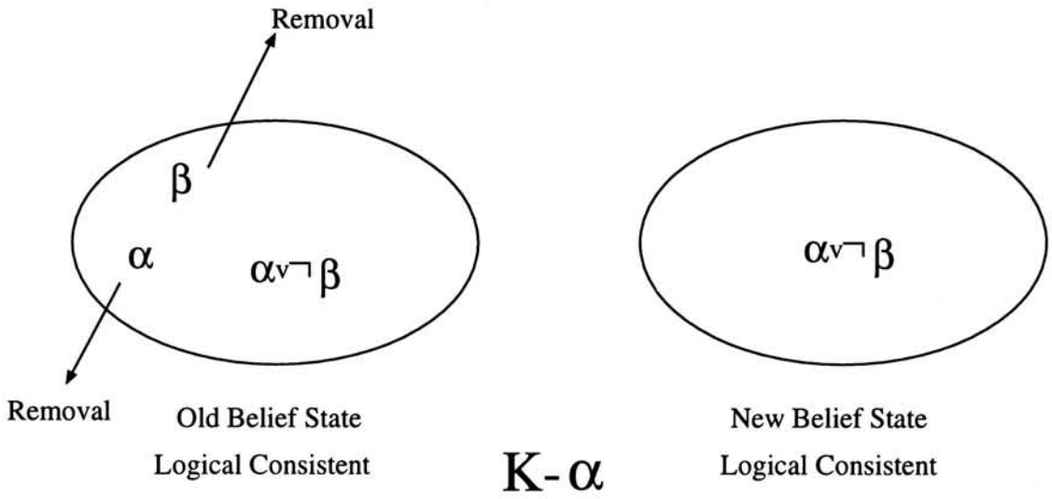


Figure 5.4: Contraction of a Belief State

However, The definition of expansion may lead to an inconsistency of a belief state as shown in Figure 5.5. Initially, $\neg\beta$ is in the belief state. When we expand it with α, the new belief state should contain α itself and its consequence. If $\alpha \vdash \beta$, the final belief is inconsistent as both β and $\neg\beta$ are present in the same state. Therefore, expansion may produce an inconsistent state.

In Chapter 7, we will show that AGM Logic is sequence dependent so that the final belief state may be different if the ordering of the operations

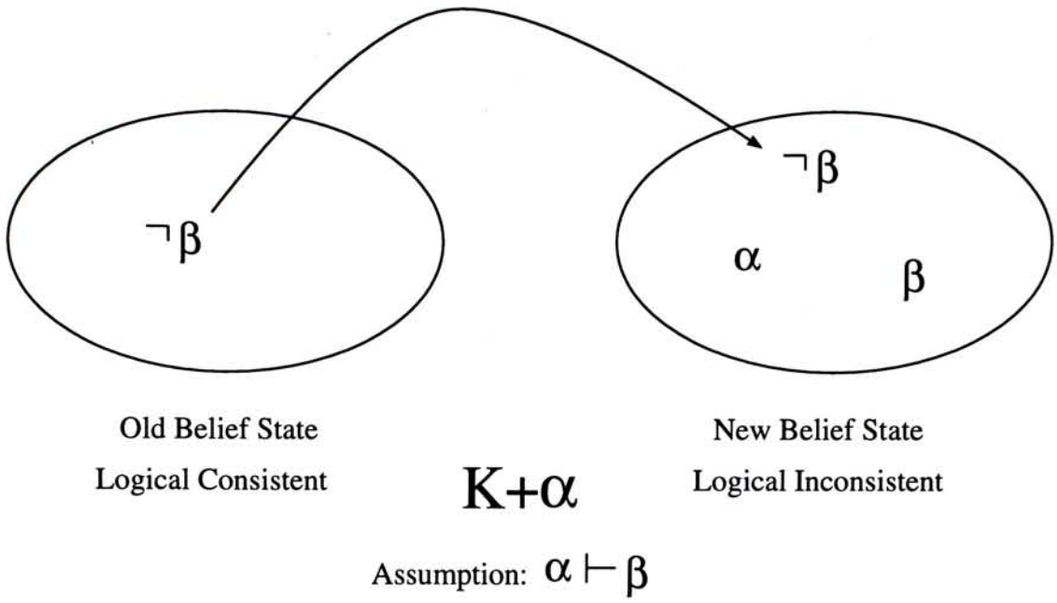


Figure 5.5: Expansion of a Belief State

changes. When we use NLBN with W-Consistency, we can always reach a unique W-Consistency final state from arbitrary initial belief state and W-Consistency is sequence independent. The details are described in Chapter 7.

5.5 Classical vs. Non-classical Logical Consistency

AGM Logic models a belief state by representing it as a belief set. It not only contains the sentences that are explicitly put into the belief state, but also the logical consequences of these beliefs. The underlying logic governs the beliefs includes *classical propositional logic.* If K logically entails Φ we write this as $K \vdash \Phi$. Where K is a set of sentences, we shall use the notation $Cn(K)$ for

the set of all logical consequences of K, i.e. $Cn(K) = \{\Phi : K \vdash \Phi\}$. Under the classical logic, whenever K is *inconsistent*, then $K \vdash \Phi$ for every sentence Φ.

NLBN with W-Consistency employs three-valued logic: t(true), f(false) and u(uncertain). Contradictory information can be present in a belief state. W-Consistency is a weaker version of consistency than the classical logical consistency. By Table 3.1, we see that there are more cases that a belief state is consistent under W-Consistency than the classical logical consistency. Although W-Consistency is not the only possible way, it opens and provides a way towards a more human-like consistency reasoning for belief representation.

5.6 Retraction vs. Suppression

AGM Logic and W-Consistency use different strategy to deal with the beliefs which are inconsistent with the revised belief state. This can be illustrated in Fig 5.6. The initial belief state S_0 contains a, b, c, d. When a new information e is added to S_0, we have to revise S_0 if a and d are inconsistent with e.

In AGM Logic, a and d are taken out from S_0 so that the new belief state S_1 do not contain them. Due to the retraction of inconsistent beliefs, the principle of recovery (postulate $(K-5)$) in Section 5.1.1 is a controversial one

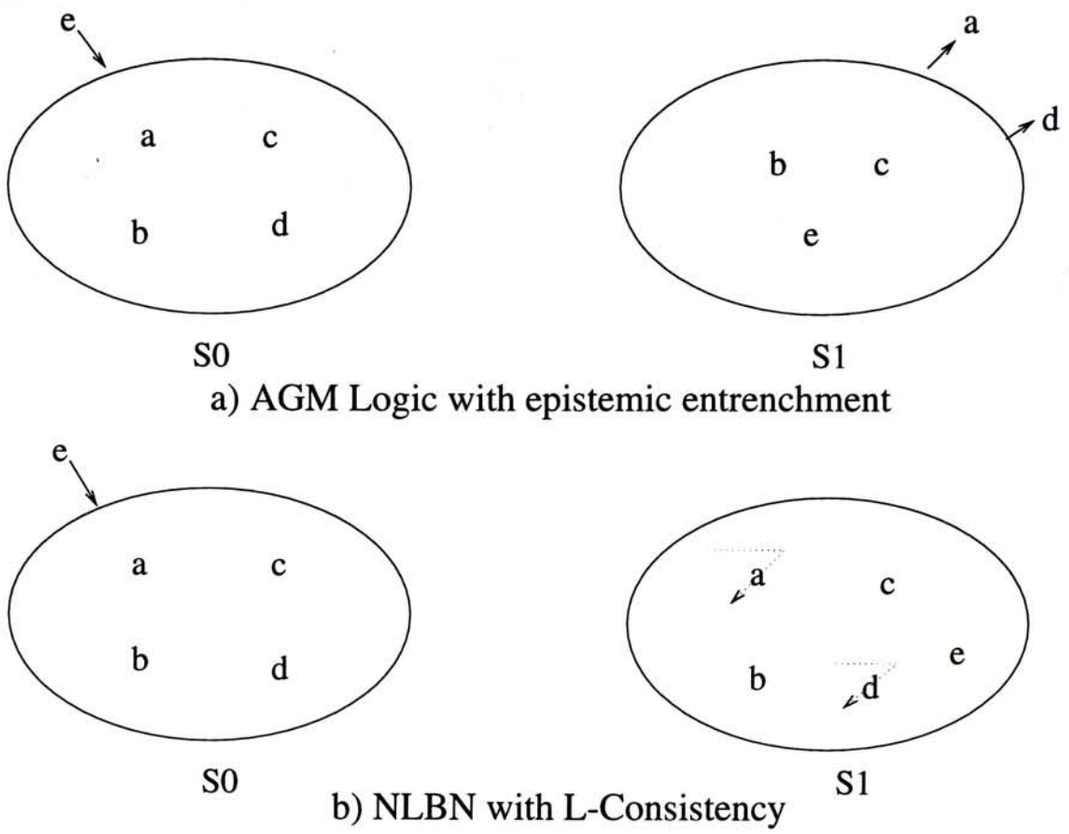


Figure 5.6: Different Strategy to Deal with Inconsistent Beliefs

in the literature ([12]). In W-Consistency, these inconsistent beliefs are only suppressed to a uncertain state and they are kept in the belief state. We can always restore these inconsistent beliefs later if it is required. Therefore, it obeys the principle of minimal change that unnecessary losses of information are avoided.

5.7 Foundation vs. Coherence Theories

There are two approaches of modeling beliefs: the foundation and coherence theory [6]. The foundation theory (e.g. TMS) states that one must always

keep the reason for beliefs. All beliefs that do no longer have a justification must be given up. If a belief state containing A is revised so that the negation of A becomes accepted, not only should A be given up in the revise state, but also all beliefs that depends on A for their justification. The coherence theory (e.g. AGM) is concerned about logical coherence and consistency of beliefs and beliefs do not require any justification. Moreover, when changing beliefs in response to new evidence, we should continue to believe as many of the old beliefs as possible.

AGM Logic is a typical system based on the coherence theory. The postulates capture some general properties of minimal changes of beliefs. However, the closure of logical consequence of AGM Logic makes implementation of a belief revision system impractical. For example, it is difficult to use in database application in which a finite representation of knowledge base is needed.

NLBN is neither purely coherence nor foundation. It imposes logical consistency (W-Consistency) to all logical expressions when consistency reasoning is carried out (coherence characteristic), but on the other hand, all its IF-THEN rules [18] can be viewed as justification network (foundation characteristic) to support reasoning, including supports to logical expressions.

Chapter 6

Comparison of W-Consistency with other Systems

Previous chapter has reviewed a famous belief revision theory, *AGM Logic with Epistemic Entrenchment* and compared it with W-Consistency with NLBN. In this chapter, we continue our reviews and comparison. First, we will take a look on G-Consistency [18] which is an another consistency defined in NLBN. By the way, S-Consistency [17], a consistency close to G-Consistency, is also reviewed. Third, we will review *Truth Maintenance System* [3] which is a part of the reasoning system. It is used to keep records of logical dependencies of propositions and maintain the consistency of the reasoning system.

6.1 G-Consistency

6.1.1 Overview of G-Consistency

G-Consistency [18] is a weaker than strict logical consistency defined in NLBN. For any logical expression Φ, it receives both direct inputs and a combined input node value from its expressions. From Section 3.1, we know that a logical expression is said to be *supported* by a set of its sub-expressions if and only if the computed input proposition-value from these sub-expressions is not $(0, 0)$ and it has the same propositional-value as the logical expression. A logical expression is *opposed* by a set of its sub-expressions if and only if the combined input propositional-value of the set of sub-expressions is neither $(0, 0)$ nor supporting the logical expression. G-Consistency is based on the above definitions:

Definition G-Consistency

Given a logical expression, when the combined input value from all its sub-expressions opposes it, it is *G-Inconsistent* (with its sub-expressions); otherwise, it is *G-Consistent.*

This means that a logical expression is G-Consistent only if the inputs of its sub-expressions are **u** or the node value has to agree with the inputs. If it is G-inconsistent, the weakest degree-of-belief value sub-expressions need only to be suppressed to $(0, 0)$ to restore consistency.

Based on the notion of G-Consistency, the consistency reasoning process using it in NLBN is defined. The details are in [18].

6.1.2 Comparison of W-Consistency with G-Consistency

Both W-Consistency and G-Consistency are motivated by human common-sense reasoning that contradictory information are not as drastically resolved as those in classical logic based systems. Therefore, unlike classical logical consistency, they allow positive and negative information of the same belief present in the same belief state. During the consistency reasoning defined using these two consistencies, no information is discarded and the belief state is restored to consistent by suppressing the weaker beliefs using non-supporting logical suppressions.

For any logical expression Ψ in a NLBN, there are nine permutations of the proposition-value of Ψ versus the combined input proposition-value from a set of its sub-expressions. Table 6.1 shows all permutations and we can see that G-Consistency is the most general consistency among three and classical logical consistency is the most restrictive one. W-Consistency implies G-Consistency but G-Consistency does not imply W-Consistency.

They do not allow modus ponens although the methods of handling it

The proposition-value of the logical expression	The combined input proposition-value proposition-value computed from a set of sub-expressions	Classical Logical Consistency	W-Consistency	G-Consistency
(0,0)	(0,0)	C	C	C
(1,0)	(1,0)	C	C	C
(0,1)	(0,1)	C	C	C
(1,0)	(0,0)	I	C or I	C
(0,1)	(0,0)	I	C or I	C
(0,0)	(0,1)	I	I	I
(0,0)	(1,0)	I	I	I
(1,0)	(0,1)	I	I	I
(0,1)	(1,0)	I	I	I

Table 6.1: All permutations of the proposition-value of a logical expression vs the combined input proposition-value

Note: C = Consistent, I = Inconsistent

are different. Under G-Consistency reasoning, modus ponens does not exist. For example, if we have the following two logical expressions in a belief state S represented by a NLBN: $\neg a$ and $\neg a \rightarrow b$. Since $b \notin S$ and $\neg b \notin S$, it is G-consistent with $\neg a \rightarrow b$ (equivalently $a \vee b$). There is no change to the belief state S. However, W-consistency reasoning avoids modus ponens by suppressing $\neg a$ to a contradictory state so that b will not be introduced into

the belief state S if $a \vee b$ is a stronger belief than $\neg a$. On the other hand, if $\neg a$ is a stronger belief, it cannot be suppressed by $a \vee b$. In this case, W-Consistency accepts it as consistent like G-Consistent.

Example: Who is the killer ? Suppose during an investigation of a murder, three persons are involved: *John*, *Peter* and *Ann*. To begin with, all of them are assumed not to be the killer and the detective has to collect evidence to find who the killer is. From their past records, he believes that it is weakly the case where Not-"*John is the killer*", it is probably that Not-"*Peter is the killer*", and it is usually that Not-"*Ann is the killer*", then he has the following beliefs in his belief state:

(1) $\neg$ *John is the killer* (weakly).

(2) $\neg$ *Peter is the killer* (probably).

(3) $\neg$ *Ann is the killer* (usually).

To simplify the propositions, let a denotes the proposition "*John is the killer*" in (1), b denotes the proposition "*Peter is the killer*" and c denotes the proposition "*Ann is the killer*". If a piece of evidence shows that *John* and *Peter*'s testimony contradict each other and both of them cannot prove any alibi. It is strongly believed that either one of them is the killer or both of them are killers, then we will have the following belief:

(4) *John is the killer* $\vee$ *Peter is the killer* (strongly).

Or, in the simplified form, $a \vee b$. If further evidence shows that *Ann*'s statements support very strongly what *John* says, it is then believed definitely that Not-"*Ann is the killer*" implies Not-"*John is the killer*", i.e. $\neg c \rightarrow \neg a$ or equivalently in a conjunctive normal form $\neg a \vee c$:

(5) $\neg$ *John is the killer* $\vee$ *Ann is the killer* (definitely).

The TAO for these beliefs is: *weakly* $<$ *probably* $<$ *usually* $<$ *strongly* $<$ *definitely.* We use W-Consistency and G-Consistency to see how they deal with this example.

First, if we use W-Consistency, after four *Add* operations for beliefs (1) to (4), the detective has the belief state S_1 in the NLBN expressed by their relative position on a TAO scale as shown in Figure 6.1. Under W-Consistency reasoning, the consistency checking procedure reveals that $a \vee b$ is inconsistent with $\neg a$ and $\neg b$. Non-supporting logical suppressions are introduced to the two sub-expressions $\neg a$ and $\neg b$ and this is shown in S_2 in the Figure 6.1. When belief (5) is added to S_2, it is W-Inconsistent with $\neg c$ because $deg[\neg c]$ is smaller that $deg[\neg a \vee c]$. Another logical suppression is introduced and the final belief state is S_4. The beliefs (1) to (3) becomes contradictory, i.e. what he knows about is either *"John or Peter, or both, are killers"* and

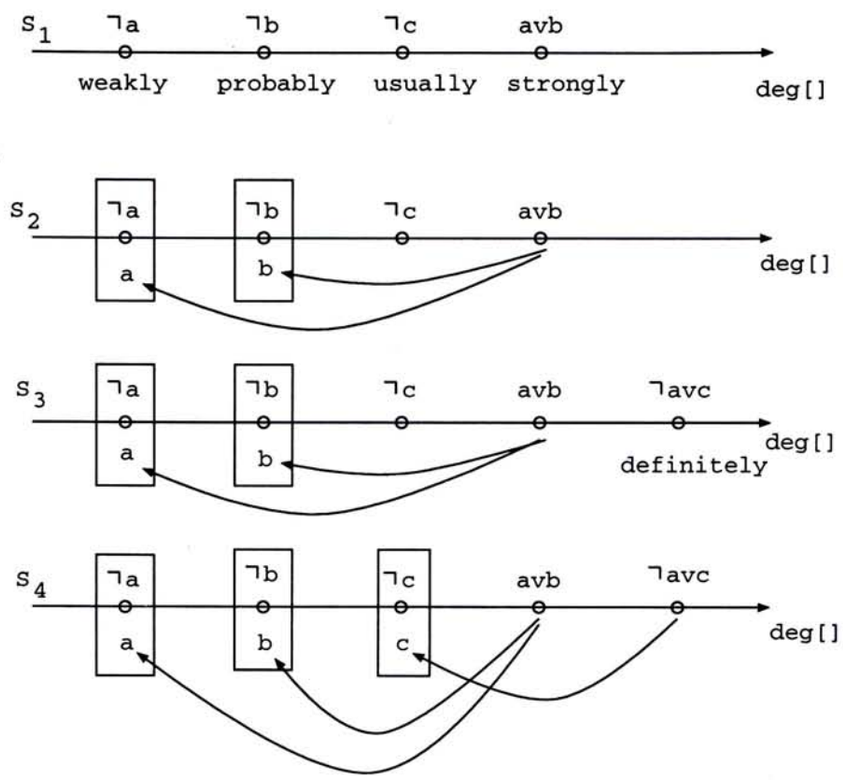


Figure 6.1: Example using W-Consistency

"Ann is ignorance implies that John is ignorance". The detective cannot conclude who the killer is from the present evidence. He has to collect more information to make the conclusion.

If the detective uses G-Consistency during the consistency reasoning. After adding belief (1) to (4) as shown in state S_1 in Figure 6.1.2. Belief $a \vee b$ is G-Inconsistent with its sub-expressions. The weaker belief a is suppressed to an uncertain state. At this time, the detective does not doubt the status of both a and b, instead, he still considers that Peter is ignorance. When belief (5) is added in, it is still G-Consistent and no change is needed. As a result,

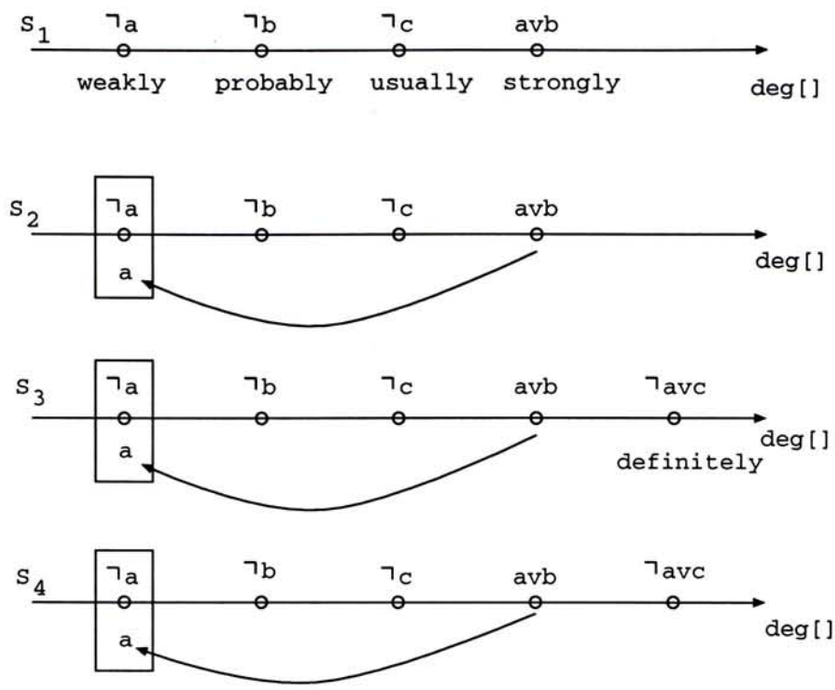


Figure 6.2: Example using G-Consistency

only the status of John is questionable as he has the weakest credibility in his testimony.

From the above example, we see that W-Consistency is very cautious and always prefers the composite belief to its sub-expressions so these sub-expressions must be suppressed. It is still possible to reason based on the composite belief but avoid to make any decisions based on one of its sub-expressions since they are not certain. It is better to wait for much more definite information (i.e. deg[sub-expression] > deg[logical expression]) to clear the uncertainty.

In Chapter 7, we use an example to show that G-Consistency is sequence dependent and different belief states may be produced from a different consistency reasoning sequence. However, W-Consistency ensures that there is a unique final belief state and sequence independent. As a result, W-Consistency is more suitable for lazy consistency reasoning than G-Consistency. The details will be discussed in Chapter 7.

6.2 S-Consistency

6.2.1 Overview of S-Consistency

S-Consistency [17] is a stronger version of G-consistency which obeys modus ponens inference rule in classical logic in some extent.

Definition S-Consistency

Given a logical expression Ψ, when there exist a unique weakest non-supporting and not non-interfering sub-expression α from Ψ where $\deg[\alpha] < \deg[\Psi]$, and, all other sub-expressions of Ψ disagree with Ψ, it is S-Inconsistent; otherwise it is S-Consistent.

This type of consistency reasoning demands that if there exists a non-supporting weakest inconsistent sub-expression, make it S-consistent by suppressing it to support the logical expression using a *supporting logical suppres-*

sion(Section 3.2). Basically, S-Consistency is a special case of G-Consistency, if it is S-Consistent then it is G-Consistent.

6.2.2 Comparison of W-Consistency with S-Consistency

Since S-Consistency is a sub-class of G-Consistency, most of the characteristics of S-Consistency are similar to G-Consistency except the logical suppressions used. Consistency reasoning process using S-Consistency is also the same as G-Consistency. For example, the weaker beliefs are suppressed and no information is discarded during consistency reasoning.

In previous section, we know that W-Consistency does not allow modus ponens. However, S-Consistency is motivated by the modus ponens inference rule. For example, if we have two logical expressions in a belief state S represented by a NLBN: $\neg a$ and $a \vee b$ and b is the unique weakest not non-interfering belief when compared to $\neg a$. To maintain S-Consistency, b has to be suppressed to $(1, 0)$ to support $a \vee b$.

When we use S-Consistency to model the "*Who is the killer?*" example in the previous section, the update procedures are shown in Fig 6.3.

When $a \vee b$ is inserted, it is S-Inconsistent with $\neg a$ and $\neg b$. Since there exist a unique weakest inconsistent disjunct $\neg a$, it is logically suppressed to

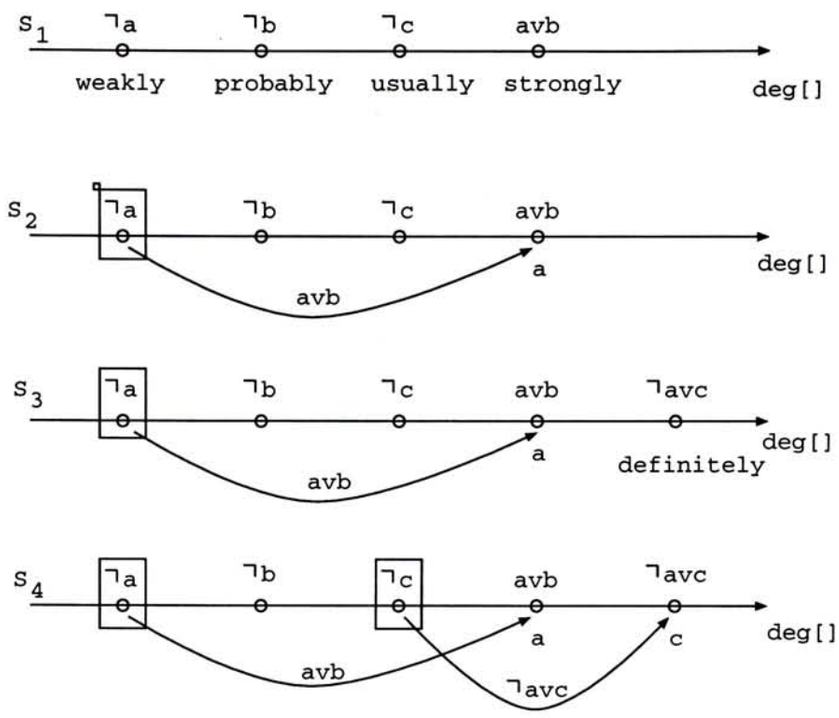


Figure 6.3: Example using S-Consistency

a with $\deg[a] = \deg[a \vee b]$. After that, $\neg a \vee c$ is added into the belief state. It is also inconsistent with its disjuncts. Therefore, the unique weakest belief $\neg c$ is suppressed to c with $\deg[c] = \deg[\neg a \vee c]$. The final belief state is that we can conclude "*Charles tells lies*", "*Elizabeth tells lies*" and "*Not Diana tells lies*". S-Consistency reasoning requires that if possible, find the most likely belief(s) that supports the belief state as a whole. This corresponds to the notion that there must always be an explanation for a problem. The result is very different from W-Consistency in which we would like to wait for more definite information before any decision is made.

6.3 Truth Maintenance Systems

6.3.1 Introduction of Truth Maintenance Systems

Truth maintenance system (TMS) [3] is part of the reasoning system that works with a *problem solver* (PS) to maintain consistency of the belief state and remove contradictions. It is a foundation system [6] where all beliefs have to be soundly justified. It supports PS's activity in the following ways:

- To record and propagate logical dependencies among propositions which are manipulated by a PS.
- To answer queries about whether a given proposition belongs to the current set of beliefs.
- To detect contradictions in the set of beliefs and resolve them.

TMS records and maintains proofs. The proofs are made up of *justifications* connecting *nodes*. Nodes represents assertions, rules, or other program beliefs. A node may have several justifications, each of which represents a different method of deriving the belief in the node. For each node, TMS computes whether or not a belief in the node has *well-founded support*, the set of non-circular proofs, of the believed nodes. The node p is *in* (or believed), that is $p \in S$ where S is the current set of beliefs, if at least one of these justifications is valid; and it is *out*, i.e. $p \notin S$, if it is without a valid justification. There are two basic form of justifications. The first is the *support-list justification*, which is of the form

(SL (*in*list)(*out*list))

A support-list justification is *valid* if each node in its *in*list is *in*, and each node in its *out*list is *out*. When both the *in*list and *out*list are empty, the justification forms a *premise* which is always valid so the node it justifies will always be believed. A node that has an empty *out*list represents *monotonic justification* which is justified by the beliefs in the nodes of the *in*list. Monotonic justification means that the validity of a proof is not affected by the addition of new beliefs. *Assumptions* are nodes with non-empty *out*list and they can be retracted by justifying any nodes in the *out*list. These justification can be interpreted by viewing the nodes of the *in*list as the reasons for making the assumption; the node of the *out*list represent the incompleteness of knowledge authorizing the assumption. Note that TMS does not take the logical meanings of the node into consideration.

The logical interpretations of propositions are lost in TMS representation. Therefore, the two views (belief or disbelief) of a proposition p must be represented by two different nodes p and $\neg p$ (the negation of p). A node representing belief p in the state *out* does not indicate that $\neg p$ is in the current set of belief. It may happen that $\neg p$ is in the current set of belief, out of the current set, or even no node representing $\neg p$ is in the belief set. The implicit inconsistency of p and $\neg p$ in any logical system is not part of TMS, and it is the duty of PS to highlight any such kind of contradiction to TMS.

The second is the *conditional-proof justification*, which is of the form

(CP (consequent) (*in*hypotheses) (*out*hypotheses))

A justification of this form is valid if the consequent node is *in* whenever each node of the *in*hypotheses is *in* and each node of the *out*hypotheses is *out*. A node justified by this form represents an implication.

6.3.2 Comparison of TMS between W-Consistency with NLBN

TMS is a dependency network management system [3] which is intended to support the problem solver's activity. It only keeps track of dependencies among propositions. They work with problem solvers to maintain belief consistency. A Neural-Logic Belief Network (NLBN) is a general belief revision and reasoning system, which has both truth maintenance and problem solving parts within a unified framework.

Unlike TMS where each belief can only be in the states of either **in** or **out**, any belief in NLBN can either be **t** (true), **f** (false), or **u** (uncertain). According to the basic set inclusion interpretations, we have classified a node p **in** to be **t**, while p **out** to be either **f** or **u**. Note that in NLBN, which follows basic logical semantics, both p and $\neg p$ are represented as a single base node, whereas in a TMS, p and $\neg p$ have to be represented separately and

whether their co-existence is contradictory will depend on how the problem solver views the case.

Both systems recognize the importance of minimal change to existing beliefs when updating the current set of beliefs. TMS achieves the principle by finding the maximal assumptions which contain those assumptions which are nearest to contradiction node and its foundations and none of which are in the foundations of another element of the set [9]. Retraction of a maximal assumption makes sure that the minimal change to existing beliefs. NLBN employs the W-consistency reasoning to resolve knowledge contradictions in the first case. This process follows the principle of minimal change to the beliefs and only the weakest relevant beliefs are affected. This avoids unnecessary removals of assumptions. Indeed, NLBN only suppresses the weakest belief(s) and does not expel them from the belief state. They remain as beliefs and will come out again when the suppression is relaxed [19]. However, in TMS, the assumptions retracted is removed from the belief set and it will not be added back again unless the problem solver does it later. On the other hand, when the contradiction node is introduced, it will remain in the belief state until the problem solver removes it.

By adopting TMS-style truth maintenance, it also inherits the risk of running into unsatisfiable circularity (i.e. circular justification) [9]. Fortunately, in a NLBN, such situation will end up as having contradiction in the

affected nodes and contradiction does not support other parts of reasoning. The unsatisfiable circularity situation cannot happen in a NLBN with the consistency reasoning process, and it always terminates with a definite, and stable belief state.

Chapter 7

Lazy Consistency Reasoning using W-Consistency

In this chapter, we describe the process of Lazy Consistency Reasoning modeled in NLBN and use one example to show the effect of update sequence on knowledge systems with different consistency restoring methods.

Although many beliefs in human mind are correlated and dependent, they can be grouped into clusters of independent sets of knowledge. For example, let us consider knowledge of making food and the knowledge of solving mathematics problems. Each of these knowledge domains consists of many beliefs. They are mostly non-related and a change of a belief in one knowledge domain does not affect the other. As a result, the inconsistency of one set of knowledge has no influence on the other domain.

Consistency reasoning process is a potentially time and resource consuming task. What we usually do is just register the belief changes and then get on with the more important and immediate reasoning tasks. This implies that the whole belief state may not be consistent - or more correctly termed: partially inconsistent. It is acceptable when the inconsistent set of knowledge does not affect those which are needed in immediate reasoning.

This type of cognitive behavior can be modeled by a NLBN with W-Consistency reasoning. We shall first allow belief updates to be carried out without a consistency reasoning process. The resulting NLBN and its associated belief state may then be partially W-inconsistent after belief changes. When a certain part of the NLBN (a set of knowledge) is required for reasoning, a lazy consistency reasoning process is then carried out starting from the most sought-after belief (the output) sequentially to determine what are the current coherent beliefs with respect to the portion of beliefs required for reasoning.

Given a node in NLBN, we can get a set of nodes which is related to it. A node i is related to another node j if there is a direct connection between each other or indirectly connected through other nodes by combinative links. Therefore, we can always extract a portion of the network S_p from S which contains the maximum number of nodes that are related and we call it *maximal related sub-network*. Obviously, different maximal related sub-network

represents different kinds of knowledge and they are independent. When a NLBN is used for reasoning, some sub-networks may be W-inconsistent while some are W-consistent as the time passes and those inconsistent do not affect the reasoning using consistent portions. When a W-inconsistent sub-network is needed for reasoning, the lazy W-consistency reasoning process is performed on it.

Since NLBN only accepts logical expression in Conjunctive Normal Form and W-consistency reasoning process terminates finitely in a consistent state, as a result, lazy W-consistency reasoning applies to any maximal related sub-network will produce a consistent state on that sub-network. Moreover, the consistency reasoning process deals with each logical expression locally and a logical suppression on any sub-expression does not affect other sub-expressions. As a result, the final belief state is unique. Detailed proofs are given in next section.

7.1 Proof of Lazy Characteristic of W-Consistency

To show that there is a unique final belief state of a NLBN no matter when the lazy W-consistency reasoning process is applied if the initial belief state of the NLBN and the update operations are the same. We need the follwoing lemmas.

Throughout this section, we use S_i $(i \geq 0)$ to represent belief states of a NLBN, u_j $(j \geq 0)$ to represent an update operation on a belief state. $S_i + u_j$ represents an update operation carried out on a belief state S_i without performing any consistency reasoning process. $S_i \rightarrow S_{i+1}$ denotes consistency reasoning process is carried out on a belief state S_i and generates a new belief state S_{i+1}. A belief state of a NLBN must have either no suppressions or at least one suppression and we call the former a non-suppressed state, denoted as S_i, and the later a suppressed state, denoted as $\overline{S_i}$.

Lemma 7 Given a NLBN with a pure disjunctive logical expression Ψ and its sub-expressions α_i $(i > 0)$, the final belief state must be unique no matter when the lazy W-consistency reasoning process is applied if the initial belief state and the update operations are the same.

Proof: Since belief states of a NLBN must be either suppressed or non-suppressed. There are four cases we have to consider:

1. All belief states are non-suppressed states, that is, all of the belief states do not have logical suppressions in them.

   There are no logical suppressions in a belief state if and only if after the belief change, it is still W-consistent. By the definition of consistency maintenance, it does not make any changes to the belief state if

it is already consistent. Therefore, consistency reasoning process can be applied at any time as it does not have any effect on the belief state. Moreover, the state of each node is just the strongest input from different inputs (direct inputs and combinative input). The sequence of update operations have no effect. Therefore, the final state must be unique.

2. All belief states are suppressed states.

   Only several update operations are allowed in order that all sequence of belief states are suppressed:

   - Update one suppressed sub-expression, α_i of a logical expression Ψ say, such that the node value is $((0,1), \deg[\alpha_i])$ where $\deg[\alpha_i] \leq \deg[\text{logical expression}]$, the node is still suppressed.
   - Update one suppressed node, α_i say, such that the node value is $((0,1), \deg[\alpha_i])$ where $\deg[\alpha_i] > \deg[\text{logical expression}]$, the suppression can be removed provided that there is at least one other sub-expression being suppressed.
   - Update one suppressed node, α_i say, such that the node value is $((0,0), \deg[\alpha_i])$ or $((1,0), \deg[\alpha_i])$, the suppression can be removed provided that there is at least one other sub-expression being suppressed.
   - Update the logical expression Ψ such that it is $((0,0), \deg[\Psi])$, all sub-expressions with degree smaller than or equal to $\deg[\Psi]$ must

be suppressed to $(0, 0)$ with their original degree.

- Update the logical expression Ψ such that it is $((1, 0), \deg[\Psi])$, all sub-expressions with $((0, 1), \deg[\alpha_i])$ where $\deg[\alpha_i] \leq \deg[\Psi]$ are suppressed to $((0, 0), \deg[\alpha_i])$.

From the above update operations, we see that as long as there is at least one node being suppressed, all other sub-expressions are changed from suppressed to non-suppressed or vice versa with the degree-of-belief values remain the strongest ones for the nodes locally. Therefore, the final belief state depends only the strongest belief of each node.

Under the definition of W-consistency, a sub-expression is suppressed if and only if it is a weakly disagreeing input or influencing input to the logical expression. The suppression is not affected by the node values of other sub-expressions. Also, a suppression to a sub-expression is relaxed if and only if the node value of that sub-expression supports the logical expression or it becomes strongly disagreeing input to the logical expression after any update operation. After the relaxation, the sub-expression restores the original node value which is the strongest belief among different inputs to the sub-expression. Moreover, only non-supporting logical suppression is allowed in W-consistency which only suppresses the required nodes to $((0, 0)$, deg[sub-expression]) where

deg[sub-expression] is the strongest degree-of-belief value of the sub-expression. As a result, the final belief state must be unqiue no matter when the lazy W-consistency reasoning process is applied.

3. The initial state S_0 and the final state S_n are non-suppressed states. However, there is at least one intermediate state $\overline{S_i}$ which is a suppressed state.

   Assume there is only one suppressed state between S_0 and S_n. That is,

   $S_0 + u_0 \rightarrow S_1, \ldots, S_{i-1} + u_{i-1} \rightarrow \overline{S_i}, \overline{S_i} + u_i \rightarrow S_{i+1}, \ldots, S_{n-1} + u_{n-1} \rightarrow S_n$.

   By (1), we can apply the consistency reasoning at any time before the belief state $\overline{S_i}$ and after $\overline{S_i}$. Therefore, we may obtain

   $S_0+u_0+u_1+\ldots+u_{i-1} \rightarrow \overline{S_i}, \overline{S_i}+u_i \rightarrow S_{i+1}, S_{i+1}+u_{i+1}+\ldots+u_{n-1} \rightarrow S_n$.

   That is, we apply only one consistency reasoning process after update operation from u_0 to u_{i-1} and another after operations from u_{i+1} to u_n. We want to show that an intermediate state $\overline{S_i}$ is indeed not a restriction to apply the lazy W-consistency reasoning process.

Under W-consistency maintenance, the non-supporting suppression only makes the suppressed node change to (0,0) with original degree of that node. Therefore, the degrees of all the nodes in $\overline{S_i}$ are still maximum. After that, u_i makes state $\overline{S_i}$ change to non-suppressed state and remove the suppressed links. The suppressed nodes are restored

to previous truth value with the degrees unchanged locally. Therefore, during transition from state S_{i-1} to S_{i+1}, each node can maintain the maximum degree and the corresponding truth value or (0,0) if suppressed. As long as the update operations are the same, the maximum degree-of-belief value of each node must be unique and the consistency reasoning process can be applied at any time without affecting the final state.

4. The initial state $\overline{S_0}$ and the final state $\overline{S_n}$ are suppressed states. However, there is at least one intermediate state S_i which is a non-suppressed state.

   Assume there is only one non-suppressed state between $\overline{S_0}$ and $\overline{S_N}$. That is,

   $$\overline{S_0}+u_0 \rightarrow \overline{S_1}, \ldots, \overline{S_{i-1}}+u_{i-1} \rightarrow S_i, S_i+u_i \rightarrow \overline{S_{i+1}}, \ldots, \overline{S_{n-1}}+u_{n-1} \rightarrow \overline{S_n}$$

   By (2), we can apply the consistency reasoning at any time before the belief state $\overline{S_i}$ and after $\overline{S_i}$. Therefore, we may obtain

   $$\overline{S_0}+u_0+\ldots+u_{i-1} \rightarrow S_i, S_i+u_i \rightarrow \overline{S_{i+1}}, \overline{S_{i+1}}+u_{i+1}+\ldots+u_{n-1} \rightarrow \overline{S_n}.$$

   That is, we apply only one consistency reasoning process after update operations from u_0 to u_{i-1} and another after operations from u_{i+1} to u_n. We want to show that an intermediate state S_i is indeed not a restriction to apply the lazy W-consistency reasoning process.

   Similarly to (3), any belief state starts from suppressed to non-suppressed, and then back to suppressed again, each node can still maintain the

maximum degree-of-belief value among different inputs of that node. If the update operations are the same, all the nodes must have the same final states as the only strongest beliefs are shown as the states. Moreover, by definition of W-consistency, whether a node is suppressed or not depends on the comparison of its degree with its logical expression and not on other factors. As a result, the final belief state must be unique also in this case.

Combination of (1) to (4), given an initial belief state and given a sequence of update operations, we can always apply the lazy W-consistency reasoning process at any time without affect the final belief state for the purely disjunctive logical expression.

□

Lemma 8 Given a NLBN with a pure conjunctive logical expression Ψ and its sub-expressions α_i $(i > 0)$, the final belief state must be unique no matter when the lazy W-consistency reasoning process is applied if the initial belief state and the update operations are the same.

Proof: Since connective AND is logically symmetrical to OR in belief networks, the proof is carried out in the same ways as that of Lemma 7 by switching context from disjuncts to conjuncts, and, replacing (1,0) for (0,1) and (0,1) for (1,0) respectively.

□

Only conjunctive normal form of logical expressions are allowed in NLBN. Therefore, we can see that the general structure of a NLBN consists of three layers of nodes: the first layer includes all literals or the name of simple propositions without any logical connectives attached, represented by α_i. The second layer includes disjunctive logical expressions, denoted by β_i and the third layer includes conjunctive logical expressions with its sub-expressions can be disjunctive logical expressions, denoted by γ_i.

Theorem 10 (The uniqueness of final belief state for lazy W-consistency resaoning)

The final belief state of a maximum related portion of a NLBN must be unique and W-consistent when lazy W-consistency reasoning is applied to that portion at any time, if the initial belief state and the update operations are the same.

Proof: consistency reasoning process always starts at the right most nodes (the third layer first),if present, and transverse backward. First, it checks whether the sub-expressions β_i (the second layer) are consistent with the logical expression γ_i (the third layer). If not, necessary logical suppressions are required to restore the consistency. Second, the sub-expressions α_i (the first layer) are checked to see whether they are consistent with β_i. If

not, the same consistency maintenance is carried out. Then that maximum related portion of netowrk is propagated forward from the first layer once again and the consistency reasoning process is performed again until that relevant portion is consistent.

While NLBN can only accept conjunctive normal form of logical expression, only the conjunctive and disjunctive logical expressions can be represented in a belief state. Also, any logical suppressions to the nodes on second layer can be considered as the update operations on those nodes in that layer which will need consistency reasoning to check the consistency. By Lemma 7 and 8, we have shown that lazy W-consistency reasoning process can be applied at any time for disjunctive and conjunctive logical expressions respectively. By Theorem 6, W-consistency reasoning process always terminate finitely in a consistent state. Combination of these, we can see that lazy W-consistency reasoning can be carried out at any time to a maximum related portion of the network and final state of that portion must be unique and W-consistent.

□

7.2 Example of Lazy Consistency Reasoning

In this section, we use an example to show the effect of update sequence using AGM logic, G-consistency and W-consistency. We adopt the same notations

of node value of NLBN in the example. (e.g. ((1,0), probably) means the proposition is believed with degree of probably).

Example

Suppose that an inspector holds two beliefs; "Peter probably is not the killer" and "John normally is not the killer":

- *a*: Peter is the killer ((0,1), probably).
- *b*: John is the killer ((0,1), normally).

After more evidence is obtained, there are two update operations on the belief state of the inspector.

1. Add a new belief "Peter was the killer OR John was the killer ((1,0), strongly)".
2. Change the degree of belief "not John was the killer" from normally to weakly.

First, we use G-consistency with NLBN [19] for reasoning. In Fig 7.1(a), we do operation (1) followed by (2). When $a \vee b$: ((1,0), strongly) is added, the belief state is G-inconsistent and a logical suppression is applied to the weaker belief *a*:((0,1), probably). Then the degree of belief *b* is changed from *normally* to *weakly* does not affect the consistency. As a result, the final belief state of Fig 7.1(a) is:

- *a*: ((0,0), probably)

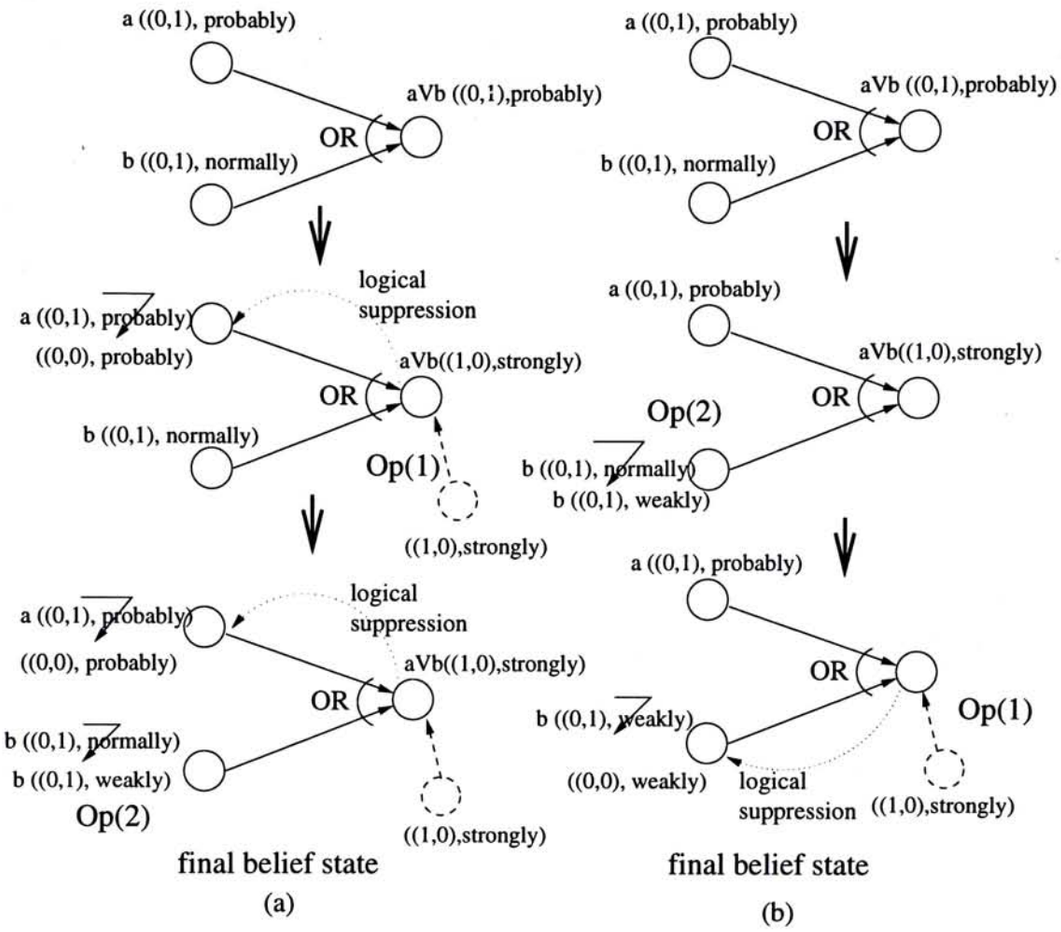


Figure 7.1: Illustration of the example using G-consistency

- b: ((0,1), weakly)
- $a \vee b$: ((1,0), strongly)

In Fig 7.1(b), we reverse the order and update the degree of b first. After that, when $a \vee b$: ((1,0), strongly) is inserted, it is G-inconsistent. The consistency maintenance procedures also suppress the weaker belief. But in this time, the weaker belief is b rather than a, so b is suppressed to an uncertain state. The final state of Fig 7.1(b) is:

- a: ((0,1), probably)

- b: ((0,0), weakly)

- $a \vee b$: ((1,0), strongly)

We see that the sequence of G-consistency reasoning process is different in the example and the final states of them are different. Therefore G-consistency is sequence dependent.

If we use AGM Logic with an Epistemic Entrenchment ordering adopted directly from the TAO of NLBN [5], the starting belief set S shall include $\neg a$ (probably) and $\neg b$ (normally) and their logical consequences. After the first update operation, $\neg b$ (normally) and $a \vee b$ (strongly) are in the resulting belief set but $\neg a$ (probably) will be expelled and a (from the closure of the belief set) will be in. After the second operation where the belief degree of b has been changed from normally to weakly, the resulting belief set shall include $\neg b$ (weakly), $a \vee b$ (strongly) and a.

If the sequence is reversed, the belief set after the second update shall include $\neg a$ (probably), $\neg b$ (weakly), and their logical consequences. If we then perform update (1), then the resulting belief set shall include $\neg a$ (probably), $a \vee b$ (strongly) and b (derived from the closure of the belief set) and their logical consequences. These results are stronger as new beliefs are brought in by the logical closure but similar to the G-Consistency reasoning in a NLBN, which are update sequence dependent.

However, using W-consistency reasoning, we can just record the belief change and do not apply the consistency reasoning immediately. The belief state becomes W-inconsistent as in Fig 7.2(a). When this set of knowledge is required for reasoning, we then apply the lazy consistency reasoning to it. Since two sub-expressions (a and b) are weakly disagreeing expressions to the logical expression ($a \vee b$). Both of them must be suppressed to (0,0). We can see that the lazy consistency reasoning process ensures a unique final belief state as shown in Fig 7.2(b) no matter what the sequence of updates is.

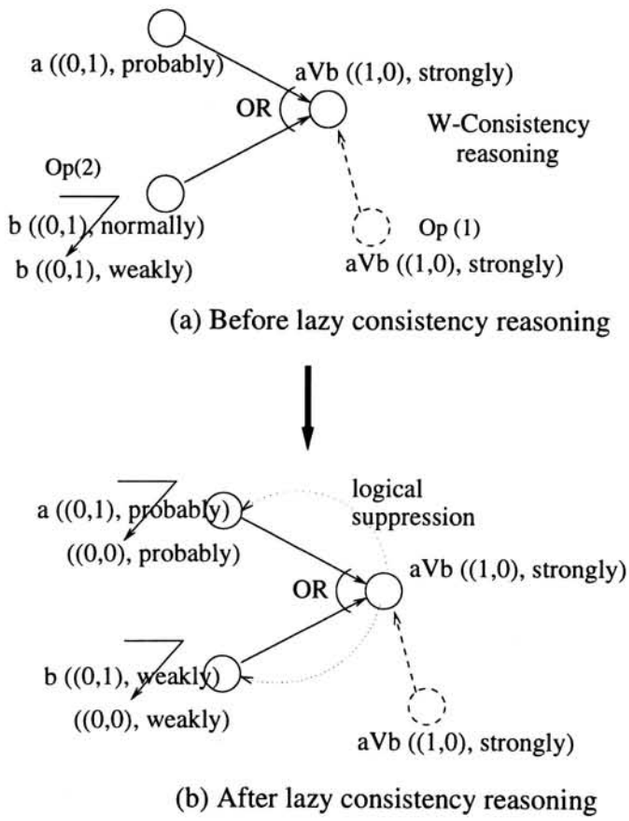


Figure 7.2: The example with W-consistency reasoning

7.3 Discussion and Application

There are two main considerations which motivate us to study this kind of behavior. First, in knowledge systems, especially for real-time applications or interactive robots, the availability of resources (space, time, etc.) are often not fulfilled. Many critical decisions have to made by sacrificing consistency of overall beliefs as long as they are not used for immediate reasoning needs. Second, in AI, modeling human reasoning is one of the utmost important tasks. Human begins show the characteristic of lazy evaluation in daily reasoning.

In this chapter, I use NLBN with W-Consistency for lazy consistency reasoning. The result is that we always have a unique final belief state with two additional respects. First, we do not need to check for consistency each time a belief state is changed. We can apply consistency reasoning at any time only when it is required. Second, we do not need to record the sequence of update operations as it does not affect the final belief state.

The direct application of Lazy W-Consistency reasoning is merging multiple knowledge bases. Knowledge bases can be sets of two-valued or three-valued propositional expressions in CNF. When producing the final consistent knowledge base, the logical expressions from the knowledge bases are added into the NLBN. This operation is equivalent to a big union ($\bigcup$) operation

on all logical expressions from different knowledge bases. We do not need to concern about the sequence of knowledge base updates and no consistency check is required. After that, a Lazy W-Consistency reasoning is applied and a unique W-Consistent knowledge set will be formed. For example, there are 2 knowledge bases with 3-valued (t, f, u): $A = \{a : (f, weakly)$, $b : (u, 0)$, $a \wedge b : (f, strongly)$, $\neg a \vee c : (t, definitely)$, $c : (t, normally)$, $d : (u, 0)\}$, $B = \{a : (u, 0)$, $b : (f, normally)$, $a \wedge b : (f, weakly)$, $\neg a \vee c : (f, definitely)\}$. When we input them into a NLBN, the initial belief state $S_0 = A \cup B = \{a : (f, weakly)$, $b : (t, normally)$, $a \wedge b : (f, strongly)$, $\neg a \vee c : (u, definitely)$, $c : (t, normally)$, $d : (u, 0)\}$. We have the ordering: $0 < weakly < normally < definitely < strongly$. After W-Consistency reasoning process, the final W-Consistent knowledge base is formed: $\{a : (u, weakly)$, $b : (f, normally)$, $a \wedge b : (f, strongly)$, $\neg a \vee c : (u, definitely)$, $c : (u, normally)$, $d : (u, 0)\}$.

If a two-valued final set of knowledge base is required, we can extract from the final NLBN those propositions with state as (1,0) or (0,1) and discard those with (0,0). For example, 2-valued mode(t, f) knowledge base with no degree-of-belief values is very common in real world applications. We can model it with proposition-value (1,0) and (0,1) and polarize degree-of-belief values to "0" and "true". If we have two knowledge bases: $A = \{\neg a, a \vee b, c\}$ and $B = \{a, \neg b, a \vee b, c\}$, we have $A \cup B = \{!a, \neg b, a \vee b, c\}$ where $!a = \{a, \neg a\}$. When W-Consistency reasoning is applied, we get a belief state of

$\{!a, !b, !(a \vee b), c\}$. If the final set of knowledge base has to be represented in two-valued mode, We discard those contradictory knowledge from the belief state and finally get a consistent knowledge base: $\{c\}$.

In the framework of 2-valued mode(t, f) with 2 degree-of-belief values: "0" and "true", W-Consistency reasoning is very cautious in the sense that as long as there is an inconsistency in a sub-expression, its logical expression (Ψ) and other sub-expressions forming (Ψ) may also set to "unknown" and expelled out to the knowledge base. For example, if we have $\{a, \neg b, a \wedge b\}$, the combined proposition value is $((0,1), true)$ while the logical expression is $((1,0), true)$. As a result, the final state of the logical expression is $((0,0), true)$. Under W-Consistency reasoning, the sub-expressions a and b must be suppressed to $((0,0), true)$ because they are influencing inputs to the logical expressions. Since all three expressions are set to "unknown" state and extracted from the final state. The final knowledge base is empty.

Chapter 8

Integration of Different Consistency Reasoning Methods

In this chapter, we try to incorporate and integrate different approaches of consistency maintenance into one unified framework so that we could allow each belief represented to adhere to appropriate approach of consistency maintenance. First, we will try to mix W-Consistency and G-Consistency into a NLBN so that either one of them can be used for a belief depending on which consistency is more suitable in its own right. We will look at the behavior of the NLBN when these two consistencies are mixed. Second, we will show that besides the its own style of reasoning, NLBN can be used to model the TMS's style of maintenance strategy.

8.1 Mixing W-Consistency and G-Consistency into a NLBN

In all previous chapters, we see that there are two kinds of consistency: W-Consistency and G-Consistency used in NLBN. We assume that either of them is uniformly used during consistency reasoning. This is not necessarily the case. We can easily integrate these two consistencies in the same NLBN. As each node in NLBN represents a logical expression, we can apply either G- or W-Consistency to each node as required. The method is to assign a flag to each logical expression indicating whether the G- or W-Consistency is to be used during consistency reasoning process. A general picture of a NLBN with mixing two consistencies is shown in Figure 8.1.

A NLBN has a finite number of base nodes, it only accepts logical expression in CNF and only three layers of base nodes and two layers of combinative-links are necessary for representing all logical expressions. Each node is assigned letter 'G' or 'W' representing that the node uses G-Consistency or W-Consistency for consistency reasoning process respectively.

For a given logical expression base node Ψ, $\{SUB_\Psi\}$ is the set of sub-expressions of Ψ. $\{G\}$ is the set of all nodes marked 'G', $\{W\}$ is the set of all nodes marked 'W'. Nodes marked 'G' are called G-nodes and those marked 'W' are called W-nodes. $\{SUPPRESS_\Psi\}$ denotes the collection of

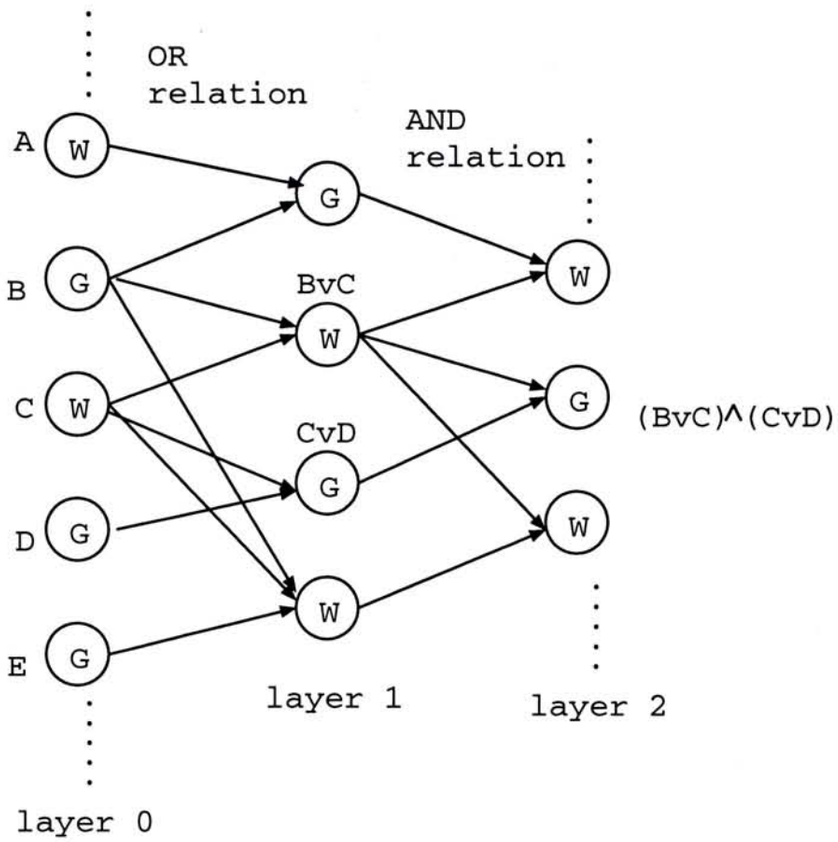


Figure 8.1: Mixing W- and G-Consistency in a NLBN

all suppressed nodes due to Ψ.

When we look at the behavior of a NLBN with two consistencies (G & W), stability and uniqueness of belief state are two main concerns. First, we expect that the consistency reasoning process still terminates finitely and the final belief state is consistent. In the following discussion, we have the assumption that consistency reasoning process must be carried out immediately when there is a belief change on a G-node and lazy consistency reasoning can be applied when W-node is changed. For example, in Figure 8.1, when we

have an update operation on node $C \vee D$, the consistency reasoning starts at once on the maximal related sub-network of $C \vee D$. However, we can start consistency reasoning at any time when the update operation is applied on W-node, e.g. one operation on $B \vee C$ and another on C before consistency reasoning.

No matter when the consistency reasoning begins, the same procedures are used and they are grouped into a function in inference engine of XHOPES called "*consistency_reasoning_ process()*" (refer to Section 4.3) shown as below:

Begin{

Consistency Reasoning starts at a node, say Ψ*;*

Collect all sub-expressions of Ψ *and form* $\{SUB_{\Psi}\}$*;*

Consistency Check to see whether Ψ *is consistent with its sub-expressions based on the type of consistency specified on* Ψ*;*

If Consistent, stop;

If Inconsistent, suppress the required nodes and put them in $\{SUPPRESS_{\Psi}\}$*;*

For each $\alpha_i \in \{SUPPRESS_{\Psi}\}$

Return to the beginning of this process and treat α_i *as* Ψ *and repeat the process.*

END }

From above procedures, we see that checking whether a logical expression is consistent or not is based on the type of consistency specified on that logical expression. It is not affected by the types of consistency on its sub-expressions. Therefore, it is carried out following the definition of that type of consistency and suppress those required to do so to resolve any inconsistencies between the logical expression and its sub-expressions. Both W- and G-Consistency requires only non-supporting suppressions (suppression that makes a belief become (0,0)). When consistency reasoning continues with the suppressed nodes, it simply treats the suppressed nodes as being updated to (0,0) and uses their specific consistency types to check with their corresponding sub-expressions (if present). As a result, consistency reasoning introduces more and more non-supporting suppressions on a NLBN. Given the local characteristics of consistency reasoning in NLBN and only non-supporting suppressions are used, combined with the fact that a NLBN contains a finite number of nodes, consistency reasoning terminates finitely when all nodes are consistent or they are exhaustively suppressed which are also consistent. The assumption of immediate consistency reasoning on G-node does not contribute any support on the stability of a NLBN, we can relax it without affecting the stability and it must terminate and consistent even all nodes (both G- and W-nodes) are lazy evaluated.

The second concern is the uniqueness of belief state of a NLBN. In Chapter 7, we see that if G-Consistency is used for lazy consistency reasoning, the

final belief state may not be unique. Therefore, when we use a NLBN mixing two consistencies for lazy consistency reasoning, the state of each node is dependent on the sequence of update operations. Even worse, the uniqueness characteristics is not also preserved when the assumption of immediate consistency reasoning on G-node is applied in a mixed NLBN. We can show this with an example in Figure 8.2.

In Figure 8.2(a), the initial belief state consists of three nodes A, B and $A \vee B$. The initial belief state is shown in Table 8.1.

Nodes	Input values	State
A	-	$((0,0),0)$
B	$((0,1)$, *normally*$)$	$((0,1)$, *normally*$)$
$A \vee B$	$((1,0)$, *strongly*$)$	$((1,0)$, *strongly*$)$

Table 8.1: Initial Belief State in Example

We have two update operations on this NLBN:

1. Add an input value $((0,1)$, *probably*$)$ to A.

2. Revise the state of B to $((0,1)$, *weakly*$)$.

In Figure 8.2, we apply consistency reasoning immediately after each operation even the updated nodes are W-nodes, the final belief state is in

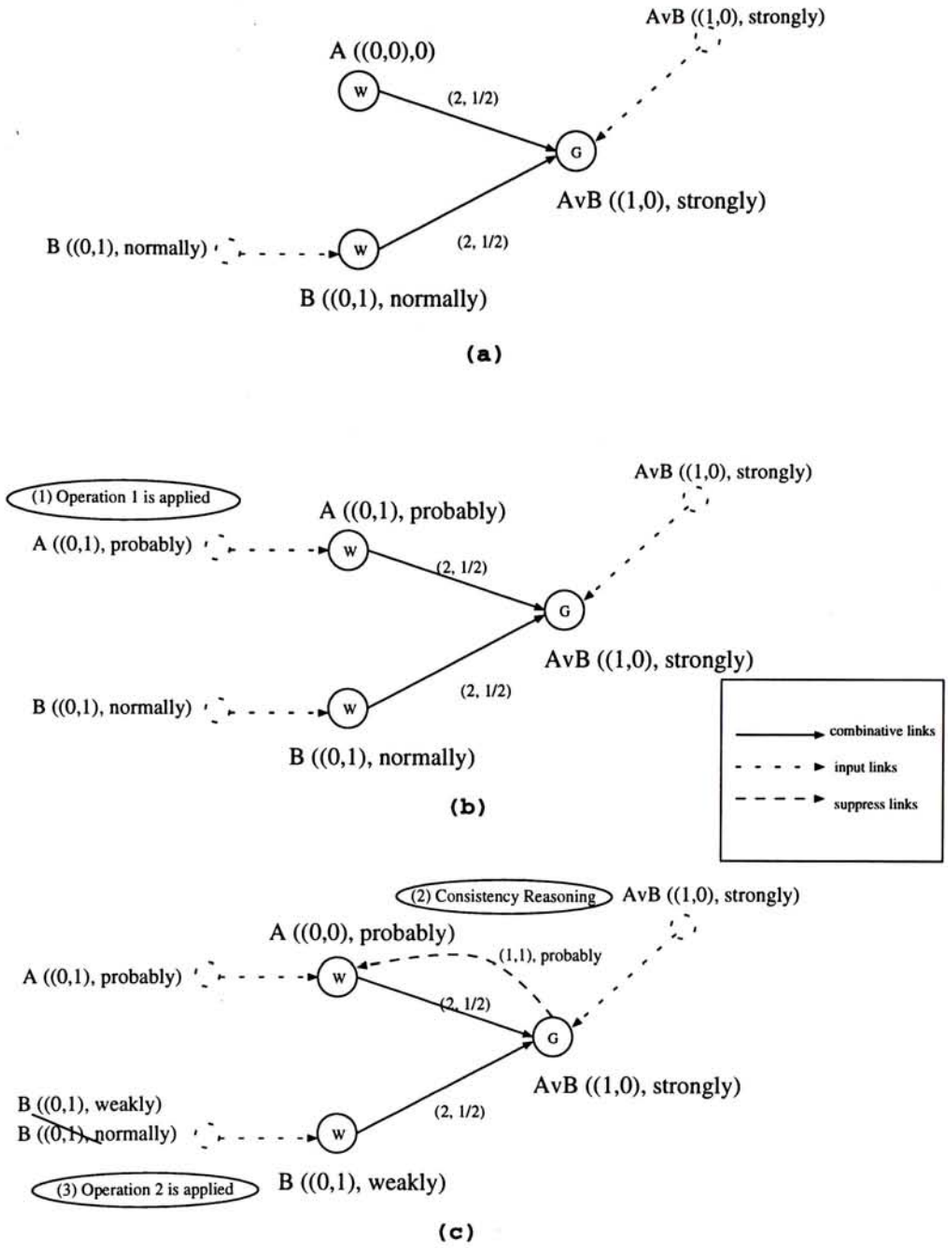


Figure 8.2: Example showing mixing of two consistencies: 1

Table 8.2.

On the other hand, in Figure 8.3, we apply lazy consistency reasoning after two update operations, the final belief state is different from previous result (Table 8.3):

Nodes	Input values	State	Status
A	$((0,1),\ probably)$	$((0,0),\ probably)$	suppressed
B	$((0,1),\ weakly)$	$((0,1),\ weakly)$	-
$A \vee B$	$((1,0),\ strongly)$	$((1,0),\ strongly)$	-

Table 8.2: Final Belief State in example

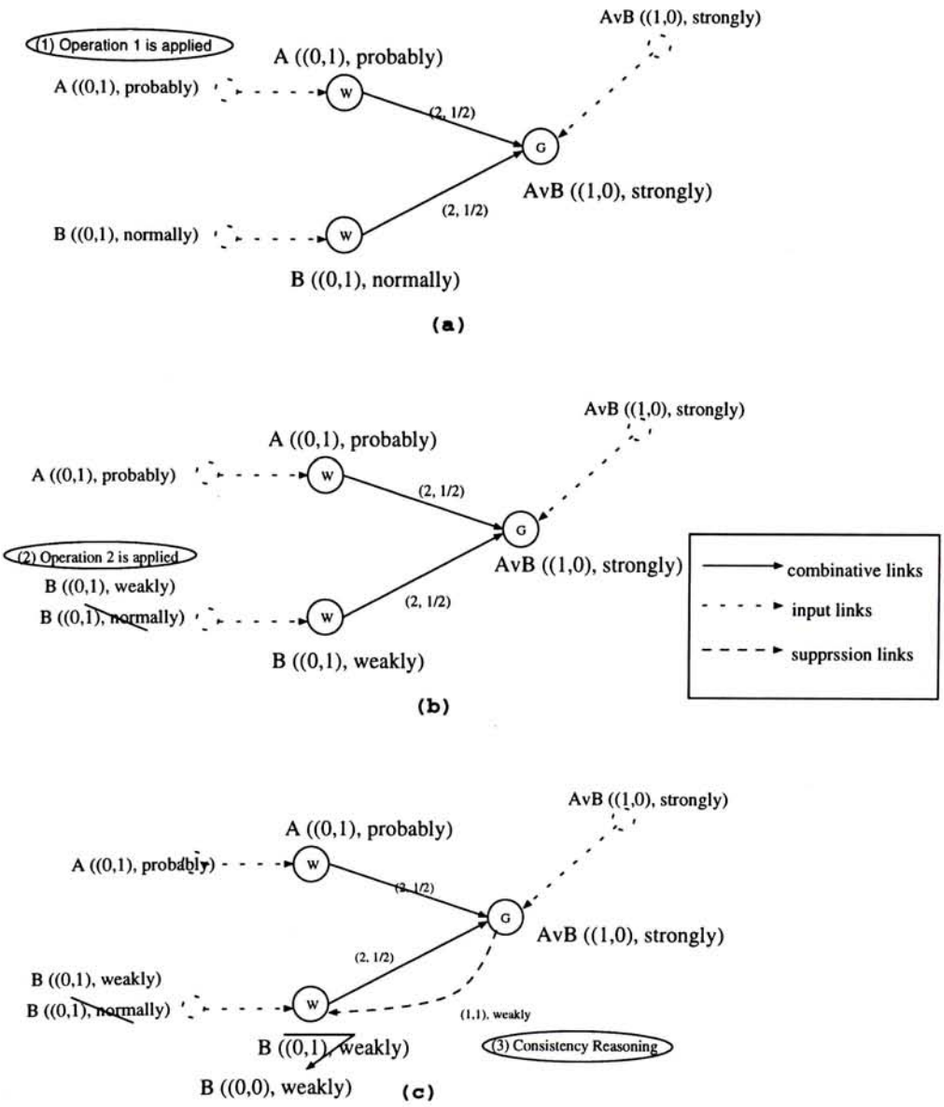


Figure 8.3: Example showing mixing of two consistencies: 2

Nodes	Input values	State	Status
A	$((0,1), probably)$	$((0,1), probably)$	-
B	$((0,1), weakly)$	$((0,0), weakly)$	suppressed
$A \vee B$	$((1,0), strongly)$	$((1,0), strongly)$	-

Table 8.3: Final Belief State for Lazy Consistency Reasoning in example

From the example, we can conclude that even though two consistencies follow their definitions to restore consistency locally for each logical-expression, the final belief state of each node may not be unique when two consistencies mixed in a NLBN. This is not what we want as we have to make more effort to keep track of the sequence of operations on a knowledge system.

Although we cannot freely apply lazy consistency reasoning in a mixed NLBN, we can still have a unique final belief state with a special procedures on NLBN. When there is an update operation on a node, say A, we have to look into the maximal related sub-network of A. If there is a G-node in it, we apply the consistency reasoning process at once no matter what kind of consistency A used. The states of all nodes in this sub-network are unique. If there is no G-node in it, we guarantee that the lazy consistency reasoning process can be applied without affecting the final belief state. In conclusion, with this arrangement, G- and W-Consistency can be integrated into a NLBN which still preserves the uniqueness of final belief state when

lazy consistency reasoning is used.

8.2 Using a NLBN for Truth Maintenance

8.2.1 TMS's Truth Maintenance Strategy

In this section, I turn to look at TMS and NLBN. First, how TMS deals with inconsistency is reviewed using an example. Then NLBN is used to handle this example using it own way. Finally, I try to model TMS-style truth maintenance using a NLBN.

The *truth maintenance process* in TMS [3] makes any necessary revisions in the current set of beliefs when a justification is added or deleted from the belief set. Retracting justifications presents no important problem beyond those of adding justifications, so we shall concentrate on the procedure for justification addition. The truth maintenance process is invoked when a new justification is added. If the new justification is invalid or if it is valid but the node is already *in*, TMS just adds this new justification without other changes. If the new justification is valid and the node is *out*, TMS records and marks this node and those affected by this node as no well-founded support. These nodes are checked to see whether any of them are valid purely on basis of the unmarked nodes. If it finds any, they are brought *in*. The marked nodes which may be affected are checked to see if they too can now

be given well-founded support. After this, TMS checks for contradictions and CP-justifications. Finally, TMS signals the PS of the changes of the status of the node.

A node in the TMS must be declared to be contradiction by the PS to signal any inconsistency. During truth maintenance, nodes are checked to see if they are marked as contradictions which mean that there are inconsistencies in the set of beliefs. To restore consistency, TMS invokes a procedure called *dependence directed backtracking* (DDB). DDB first traces backwards to find all the well-founded support of contradiction nodes that cause the inconsistencies. One point must be notified is that DDB cannot retract *premises* of monotonic justification. It presumes that all contradictions are due to the presence of *assumptions*, and searches only for set of assumptions. One of the assumptions, called *culprit*, underlying the contradiction have to be retracted to remove the contradiction. It is accomplished by providing new justification to any node, called *elective* in the *out* list of *culprit* to make the *culprit out*. In essence, the first step of DDB finds the foundations of the contradiction node C to make up the set $S = (A_1, A_2, ..., A_n)$, which are assumptions contribute to the contradiction node C. It then creates a new node NG, called *nogood*, to represent the inconsistent set S. It then justifies NG with

(CP C S ( ))

After that, it selects an A_i, the *culprit*, from S and chooses D_j from the set $(D_1, ...D_k)$ which is the set of the out nodes in the *out* list of A_i, and justify D_j with

$$(\text{SL } (\text{NG } A_1...A_{i-1}A_{i+1}...A_n)(D_1....D_{j-1}D_{j+1}...D_k))$$

This will ensure that A_i will no longer be valid when NG and other assumptions are believed. As a result, C will become *out* due to lack of enough justifications. Let's look at an example from [3]. Consider a program scheduling a meeting, to be held preferably at 10am in either room 813 or 801. At first, the beliefs in TMS are :

Node	Justification	Status
N1: Time(M) = 1000	(SL ( )(N2))	in
N2: Time(M) $\neq$ 1000		out
N3: Rm(M)=813	(SL ( )(N4))	in
N4: Rm(M)=801		out

Suppose previously scheduled meeting from PS rules out the combination of time and room for the meeting. TMS adds a new node and declared it as contradiction to represent this:

N5: CONTRADICTION (SL (N1 N3)( )) — in

To resolve the contradiction, TMS invokes the DDB process. DDB finds that N1 and N3 are two maximal assumptions causing the inconsistency. It creates N6, called the nogood node:

N6: Nogood N1 N3 (CP N5 (N1 N3)( )) — in

As there are two assumptions, TMS arbitrarily selects one of them, say N3, as the culprit and justifies N3's only node in its *out*list, that is N4:

N4: Rm(M)=801 (SL (N6 N1)( )) — in

N4 is *in* because N6 and N1 are *in*. As a result, N3 becomes *out* and the contradiction node N5 is removed. The new belief state of TMS will be:

Node	Justification	Status
N1: Time(M) = 1000	(SL ( )(N2))	in
N2: Time(M) $\neq$ 1000		out
N3: Rm(M)=813	(SL ( )(N4))	out
N4: Rm(M)=801	(SL (N6 N1)( ))	in
N5: CONTRADICTION	(SL (N1 N3)( ))	out
N6: Nogood N1 N3	(CP N5 (N1 N3)( ))	in

At this moment, another rule from PS determines that Room 801 cannot be used after all and creates another contradiction node to force a different choice of room:

N7: CONTRADICTION (SL (N4) ( )) — in

N8: Nogood N1 (CP N7 (N1) ( )) — in

DDB traces backward to find the assumptions and N1 is the only one. It creates the nogood node N8 and justifies node N2 which is the only node of *out*list of N1. The loss of belief in N1 carried N5 away as well. The final state of the belief set is:

Node	Justification	Status
N1: Time(M) = 1000	(SL ( )(N2))	out
N2: Time(M) $\neq$ 1000	(SL (N8)( ))	in
N3: Rm(M)=813	(SL ( )(N4))	in
N4: Rm(M)=801	(SL (N6 N1)( ))	out
N5: CONTRADICTION	(SL (N1 N3)( ))	out
N6: Nogood N1 N3	(CP N5 (N1 N3)( ))	in
N7: CONTRADICTION	(SL (N4)( ))	out
N8: Nogood N1	(CP N7 (N1) ( ))	in

8.2.2 Consistency Reasoning style of NLBN

Using the two principles of consistency reasoning of NLBN in Chapter 3 and the definition of G-Consistency, we look at the previous scheduling problem and examine how NLBN reasons about the previous scheduling problem using the consistency reasoning process.

At first, we have to add node N1 to N4 into NLBN (For simplicity, we use N1 to N4 to represent their respective propositions). We interpret the justifications for N1 and N3 as two rules as shown in figure 8.4a. Now, N1 and N3 are believed because N2 and N4 are in the state of **unknown** respectively. After that, we know that both N1 and N3 cannot be both believed at the same time and NLBN creates this contradiction node (N1 $\wedge$ N3) with

a node value of ((0,1),true) by using a input link. The initial state of the NLBN is shown in figure 8.4a.

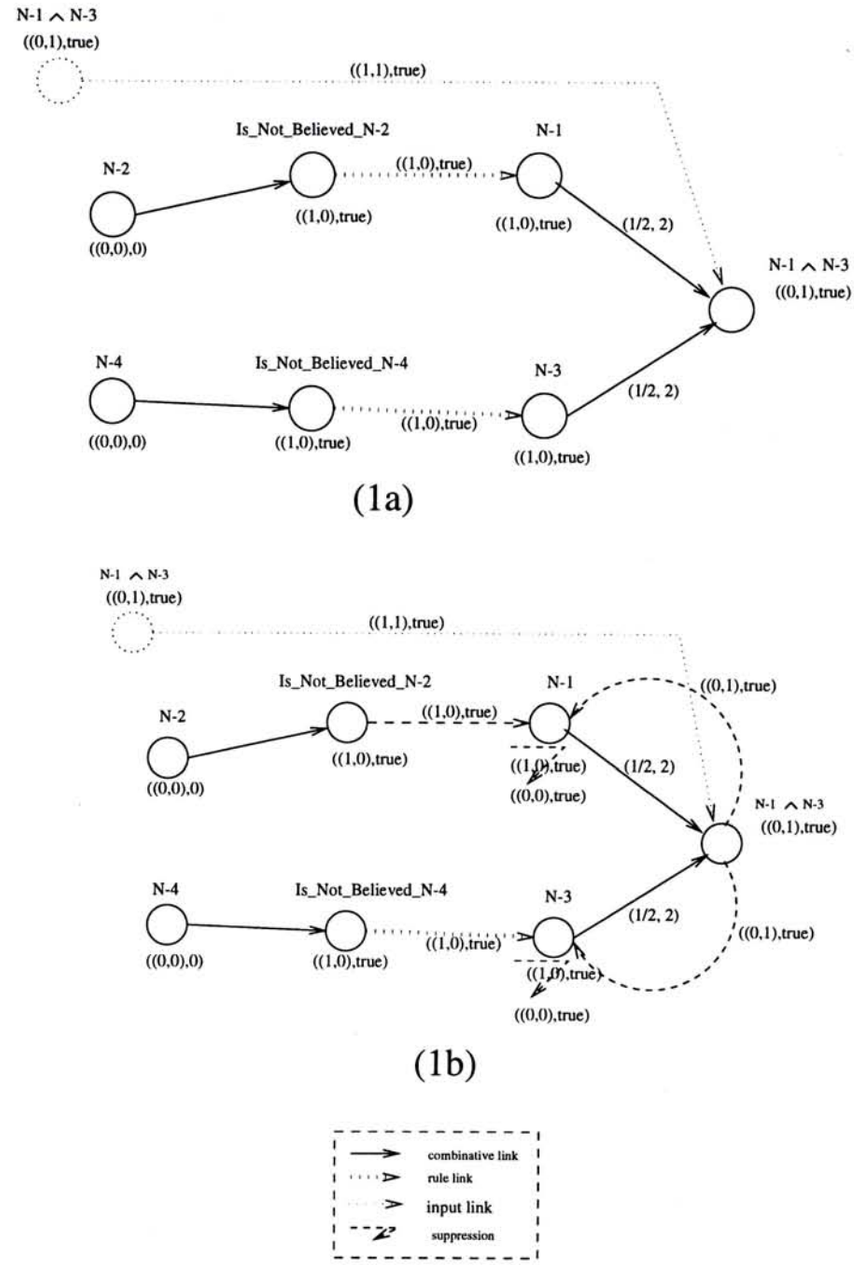


Figure 8.4: Belief states of NLBN of example 1

When the contradiction node is added, the consistency reasoning checks that it is G-Inconsistent because the combined proposition value of (N1 ∧

N3) is ((1,0),true) which opposes the external direct input value ((0,1),true). As a result, both N1 and N3 will be logically suppressed to ((0,0),true) by using an input of (0,1) from (N1 $\wedge$ N3) with the same degree-of-belief value of N1 and N3. Finally, the belief state is consistent and we conclude that "Time(M)=1000 (N1)" and "Rm(M)=813 (N3)" are contradictory. The belief state after removing contradiction is as shown in Figure 8.4b. We can see that the status of both N2 and N4 remain unchanged at an **unknown** state. As a result, the second contradiction occurred in example 1 is not an issue when it is reasoned with an NLBN.

8.2.3 Using NLBN for TMS-style Truth Maintenance

This section uses NLBN to model the truth maintenance process in the framework of TMS. [21] The basic status of a belief (node) p in TMS is either **in** or **out** (i.e. $p \in S$ or $p \notin S$ respectively, where S is the belief state). Let us interpret p **in** as a **t** for node p and p **out** as a **t** for Is_Not_Believed_p (i.e. p is either **f** or **u**) in a NLBN.

For a SL-justification in TMS, P (SL $(i_1, i_2, ...)(o_1, o_2, ...)$), it has the implied meaning that "If $i_1, i_2, ...$ are believed and $o_1, o_2, ...$ are not believed, then imply that P is believed." We can model it as the following general form with a IF-THEN rule in NLBN:

$$(i_1 \wedge i_2 \wedge ... \wedge \text{Is\_Not\_Believed\_}o_1 \wedge \text{Is\_Not\_Believed\_}o_2 \wedge ...) \mapsto P$$

In TMS, contradiction between any two beliefs has to be explicitly indicated using a CONTRADICTION node by the problem solver (PS). The DDB process of TMS will then construct a "nogood" node with CP-justification to help resolve (unjustify) some inconsistent nodes. A CP-justification in TMS can be interpreted as a belief which is always true (i.e. **in**) and it is used to justify another belief to remove the contradiction. This can be modeled by creating a TRUE node in NLBN with node value ((1,0),true) that will always make the nogood node **t**.

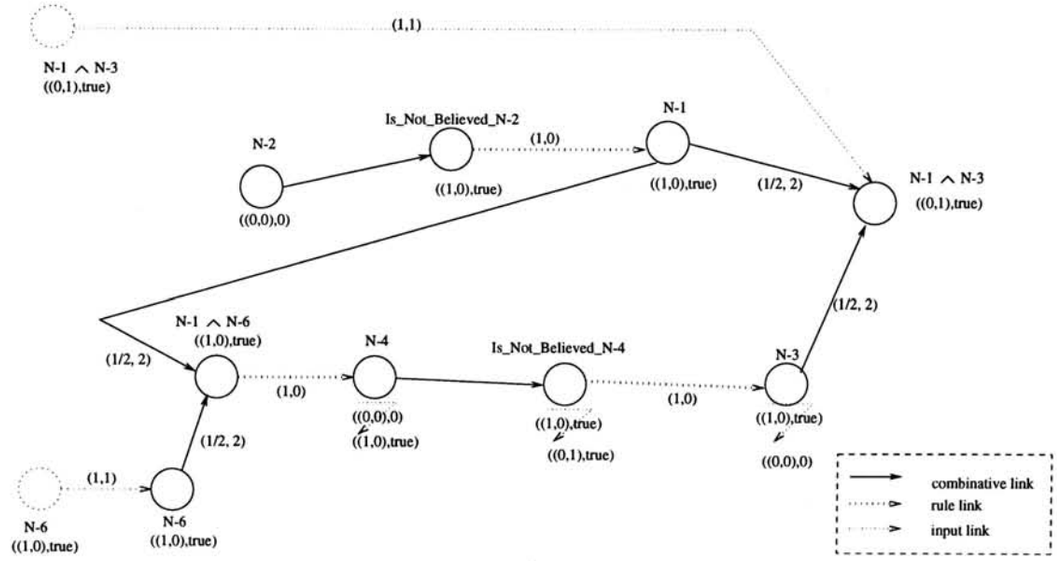


Figure 8.5: Graphical representation after removing the contradiction (N1 ∧ N3)

We shall follow the procedures in TMS for other parts of truth maintenance. Since there is no degree-of-beliefs in TMS, we polarize it to a binary "0" and "true" in NLBN for TMS-style truth maintenance.

Now, let us use the same example to show how NLBN model the process in TMS. The initial state of NLBN is the same as before as shown in figure 8.4a. The TMS-style justifications for N1 and N3 are modeled by rule-links and 'Is_Not_Believed' relations in NLBN. The beliefs "Time(M)=1000 (N1)" and "Time(M) $\neq$ 1000 (N2)" are represented by two nodes in NLBN. The same applies to N3 and N4. When we know that there is a conflict between N1 and N3 from the problem solver, a contradiction node (N1 $\wedge$ N3) with node value ((0,1),true) is created which means $\neg$(Time(M)=1000 $\wedge$ Rm(813)). The inconsistency occurs when the combined proposition-value of (N1 $\wedge$ N3) from its sup-expression N1 and N3 (i.e. a proposition-value of ((1,0),true)) opposes that of the logical expression. It traces backward and finds that both N1 and N3 are assumptions. It randomly selects one of them (the TMS strategy) to be the culprit, say N3. The nogood node, N6, is created with node value ((1,0),true) (i.e. **in**). This nogood node (**in**) and another assumption N1 are used to justify N4. Since both N1 and N6 are **t** (i.e. **in**), N4 becomes **t** by the IF-THEN rule. Is_Not_Believed_N4 becomes **f** and this makes N3 **u** (**out**). At this stage, NLBN is consistent as well as G-Consistent. Figure 8.5 shows the belief state after restoring the consistency. The result is that we choose room 801 at 10am instead of using room 813, which is exactly the same result as TMS in Section 8.2.1.

After that, if we know that Rm 801 cannot be used and N4 causes a contradiction. We create the contradiction node N7 with node value ((0,1),true)

and add a rule link between it and N4. After adding the link, the consistency reasoning process finds that it is inconsistent. NLBN traces backward and finds that N6 is not an assumption because it is always believed. N1 is the only assumption. To disbelieve N1, we add the nogood node N8 with node value ((1,0),true) (i.e. **in**). This makes N2 **t** (**in**) and N1 becomes **u** (**out**). Figure 8.6 shows how to resolve the inconsistency when N7 is created. Finally, NLBN makes N4 **u** (**out**) to restore the consistency. The result is that we have to choose room 813 again but the time cannot be 10am, which is the same as Section 8.2.1.

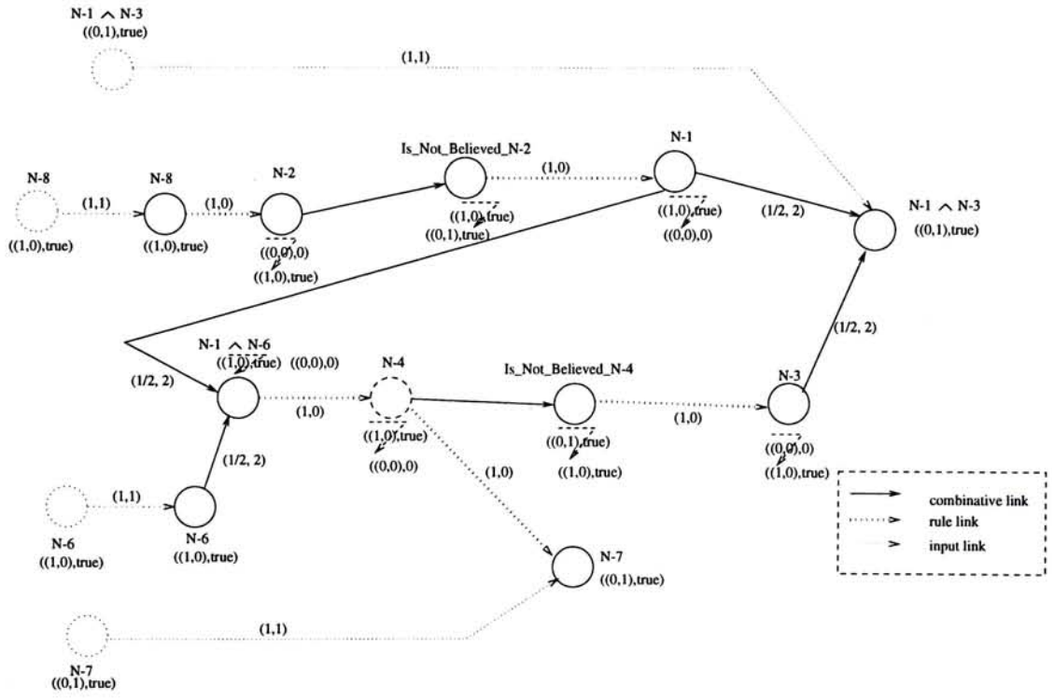


Figure 8.6: Graphical representation after removing the contradiction N7

By modeling SL-justifications as IF-THEN rules, and by resolving CON-

TRADICTION nodes by justifying nogood nodes, NLBN can achieve TMS-style truth maintenance.

8.2.4 Discussion

In the previous section, it has been shown that in addition to NLBN's own inconsistency reasoning, we could also model TMS-style truth maintenance within the same framework. When using NLBN's consistency reasoning style to resolve inconsistency, as indicated by the example in Section 8.2.2, we have to accept results that may contain contradictory beliefs in the belief states. This is due to the suppressing approach adopted in a NLBN. If we do not prefer to lose information arbitrarily, then this conforms with human reasoning that we do not always want to derive any conclusion at a particular moment when there are inconsistencies. If we cast NLBN in TMS-style truth maintenance, as discussed in Section 8.2.3, we could work out for some ways of invalidating the assumptions of a justified belief to resolve the contradiction. In the example, only a binary 0–true degree-of-belief values are used and an arbitrary selection process (adopted from TMS [3]) is employed to pick the culprit belief to be sacrificed. If we allow the full spectrum of degree values in NLBN, the TMS-like truth maintenance process could use this as a guide in picking the culprit belief from the weakest one(s) according to the TAO.

Given a NLBN, it is now clear that we could use either the consistency reasoning (G and/or W) or TMS-style truth maintenance to resolve inconsistencies. As NLBN is a flexible and accommodative belief system, we could allow each belief represented to adhere to either approach of consistency maintenance. That is, when a contradiction occurs, if a consistency maintenance approach has been designated, the system only invokes that particular process to resolve the inconsistency. This flexibility is certainly true in human commonsense reasoning where strict logical consistency reasoning is applied to more restrictive and well defined domains such as mathematics, but a more relaxing view is applied when resolving more general knowledge.

Chapter 9

Conclusion

It is believed that human beings resolve contradictory beliefs by comparing the relative strengths of the beliefs and retaining the weaker ones (in the knowledge bases by some kind of suppression) in order to achieve knowledge consistency. Inconsistent belief state does not hinder us to reason in a rational way. It is also believed that flexible human thinking does not only have one kind of consistency reasoning to resolve all inconsistent beliefs and do not always immediately maintain consistencies of our belief states in view of a belief change. Based on this philosophy, this thesis defines a weak notion of consistency (W-Consistency) and proposes a way of reasoning about inconsistencies in an inference network called NLBN. This notion of consistency is a three-valued representation and it is weaker form than the classical logical consistency. It tolerates contradictory information in the same belief set. The consistency reasoning process follows the principles of minimal change to the beliefs.

Under the formalism of NLBN, the update operations and the consistency reasoning are clearly two independent processes. We can model the characteristic of lazy consistency reasoning by incorporating W-Consistency into NLBN. The end result is that we need not do any consistency reasoning immediately after a belief change. We can apply it at any time as it is required because we can always obtain the same final consistent belief state.

The sequence independent characteristic of W-Consistency makes it suitable to combine multiple possibly contradictory knowledge bases to form a consistent knowledge base. This is carried out by adding all beliefs in these knowledge bases into a NLBN, performing a Lazy W-Consistency reasoning, and then extract the set of consistent knowledge base from the belief state of the resulting NLBN. One of the advantage of using W-Consistency is that the final knowledge base must be unique. This is achieved by suppressing those potentially inconsistent knowledge so that they do not have effect on the knowledge base. However, if any decisions made heavily depends on the timing of the update sequences (dynamic information) of the knowledge base (e.g. a robot moves and interacts with its surrounding, it must react immediately when new information is received) , lazy W-Consistency reasoning is not suitable for merging these knowledge bases because it ignores any information about timing of update operations ,and, decisions or conclusions may not be suitable with respect to the update sequence. On the other hand,

lazy W-Consistency reasoning is very useful for knowledge bases in which any conclusions depends on the states (static information) of knowledge or the update sequence does not have important impact on those conclusions (e.g. several databases in different departments of a corporation are combined to produce a final database).

After several reviews and comparisons of W-Consistency with other systems (AGM Logic with epistemic entrenchment, G-Consistency, S-Consistency and TMS), we have succeeded in blending both G- and W- Consistency for a knowledge base in a NLBN. However, the price is that we loose the full Lazy consistency reasoning capability and consistency reasoning has to be carried out after each belief change if any G-Consisent belief is involved. Also, we could use either the consistency reasoning or TMS-style truth maintenance to resolve inconsistencies.

We can foresee that not only these approaches of consistency maintenance can be incorporated into NLBN, other modes of consistency maintenance, provided that they can be modeled by NLBN, shall be able to co-exist. This deserves further studies. On the other hand, we can extend NLBN to first-order predicate level which allows us to lift knowledge from plain propositions up to abstract predicated form where we can reason about knowledge with variables.

Bibliography

[1] N. D. Belnap. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logics.* In J. M. Dunn and G. Epstein, editors, 1975.

[2] Y.M. chiu, S.H. Kwan, Y.L. Leung, W.T. Wong, and Y.F. Yam. Thesis i/ii: Hybrid knowledge systems shell hopes, 1995. Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong.

[3] Jon Doyle. A truth maintenance system. *Artifical Intelligence*, 12(3):231–272, Nov 1979.

[4] Priest. G *eg al.* (Eds.). Paraconsistent logic: Essays in the inconsistency, 1989. Philosophia Verlag.

[5] Peter Gärdenfors. *Knowledge in Flux - Modeling the Dynamics of Epistemic States.* MIT press, 1988.

[6] Peter Gärdenfors. The dynamics of belief systems: Foundations vs. coherence theories. *Revue Internationale de Philosophie*, 172:24–46, 1990.

[7] Matehew L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, 1985.

[8] J. Halpern. Reasoning about knowledge: an overview. In *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*, 1986.

[9] C. J. Petrie Jr. Revised dependence-directed backtracking for default reasoning. In *Proceedings AAAI-87*, pages 162–172, 1987.

[10] Michael Kifer and Eliezer L. Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, 9:179–215, 1992.

[11] Stephen Cole Kleene. *Introduction to Metamathematics.* North Holland, Amsterdam, 1964.

[12] Isaac Levi. *Fixation of Belief and its Undoing.* Cambridge University Press, 1991.

[13] Vladimir Lifschitz, editor. *Benchmark Problems for Formal Nonmonotonic Reasoning.* Stanford University, Stanford, Ca94305, USA, 1988.

[14] Fangzhen Lin. Reasoning in the presence of inconsistency. In *Proceedings of the 6th National conference of the American Association for Artificial Intelligence*, 1987.

[15] Jinxin Lin. A semantics for reasoning consistently in the presence of inconsistency. *Artificial Intelligence*, 86:75–95, 1996.

[16] Boon Toh Low. A network for closed-world default reasoning. In *Australian and New Zealand Conference on Intelligent Information Systems*, pages 117–122. Perth, Western Australia, 1995.

[17] Boon Toh Low and Norman Y. Foo. Consistency reasoning in neural logic belief network. Technical report, Basser Department of Computer Science, University of Sydney, NSW 2006, 1993.

[18] Boon Toh Low and Norman Y. Foo. A network formalism for commonsense reasoning. In *Proceedings of the sixteen Austrialian Computer Science Conference*, pages 425–434, 1993.

[19] Boon Toh Low and Norman Y. Foo. A weaker notion of consistency for commonsense reasoning. In *Proceedings of the Seventeen Annual Computer Science Conference.* Austrialian Computer Science Communications, 1994.

[20] Boon Toh Low and Ying Kit Wong. Consistency reasoning in knowledge bases. To be submitted to *IEEE Transaction on Knowledge & Data Engineering"*, 1997.

[21] Boon Toh Low and Ying Kit Wong. Truth maintenance using an inference network. To be submitted to *IEEE Transaction on Systems, Man and Cybercity*, 1997.

[22] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artifical Intelligence*, 28:171–172, 1980.

[23] Marvin Minsky and Seymour Papert. Perceptrons: An introduction to computational geometry. In *MIT Press, Cambridge, Massachusetts*, 1969.

[24] Abhaya C. Nayak, Norman Y. Foo, Maurice Pagnucoo, and Abdul Sattar. Changing conditional beliefs unconditionally. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, 1996.

[25] G. Priest. Logic of paradox. *Journal of Philosophical Logic*, 8:219–241, 1979.

[26] R. Reiter. A logic for default reasoning. *Artificial Intelligence J.*, 13:81–132, 1980.

[27] David E. Rumelhart. Parallel distributed processing: Explorations in the microstructure of cognition. In *MIT Press, Cambridge, Massachusetts*, 1986.

[28] John Slaney. The implications of paraconsistency. In *Proceedings of the 12th IJCAI'91*, pages 1052–1057, Sydney, Australia, 1991. Morgan Kaufmann.

[29] Geoffrey G. Towell and Jude W. Shavlik. Refining symbolic knowledge using neural networks. In *International Workshop on Multistrategy Learning*, 1991.

Appendix A

Test Case for Merging Knowledge Bases Using XHOPES

The objective of this test case is to show the uniqueness of final state by merging multiple knowledge bases regardless of the sequences of merging and how lazy the consistency reasoning process is carried out. In this test case, we use XHOPES, described in Chapter 4, as a tool for merging three knowledge bases: A, B and C. The consistency type used is W-Consistency. Each knowledge base contains about 40-50 facts (logical expressions), which are arbitrarily generated (Table A.1). Moreover, we polarize the degree-of-belief values to "0" and "*true*" only since specturm of degree-of-belief values do not affect the algorithm of merging and consistency reasoning process. As a result, all facts listed in the knowledge bases have degree-of-belief values "*true*".

After generating 3 knowledge bases, we combine them in several different ways:

1. $((A + B) + C)$

2. $((B + C) + A)$

3. $(A) + (B) + (C)$

4. $(A + B + C)$

where $A + B$ means A and B are combined (union of A and B), (A) means that W-Consistency reasoning has been applied to knowledge base A and it must be W-Consistent.

For example, for $((A + B) + C)$, We can use ADD command to insert facts into XHOPES. $A + B$ can be achieved by adding all facts in A and B. After that, W-Consistency reasoning is applied to produce $(A + B)$. Facts in C is then inserted followed by another W-Consistency reasoning and finally $((A + B) + C)$ is obtained. After testing 4 cases, the final knowledge bases are the same as shown in Table A.2.

From Table A.2, we see that most of the facts in the final knowledge base have states of "*unknown*". It is reasonable as facts in three knowledge bases are arbitrarily generated. Many of them are inconsistent so that after W-Consistency reasoning, those inconsistent knowledge will become "*unknown*" (or "*contradictory*").

In this test case, we see that a unique final knowledge base can be obtained when we change the sequences of merging and the time W-Consistency reasoning applied.

KB A	KB B	KB C	KB A	KB B	KB C
a	$\neg a$			$m \vee \neg b$	
$\neg b$	$\neg b$			$n \wedge t$	
c	c	c		$k \vee w \vee p$	
d	d	d		$\neg x \wedge l \wedge e$	
	e	$\neg e$			$\neg b \vee \neg w$
$\neg f$	$\neg f$	$\neg f$			$\neg q \wedge \neg m$
		$\neg g$			$\neg e \vee \neg m$
h	h	$\neg h$			$z \wedge \neg i$
$\neg i$	i	i			$\neg s \vee c$
$\neg j$	$\neg j$	$\neg j$			$p \vee i$
k	k				$t \wedge \neg c \wedge b$
l	$\neg l$	l			$n \wedge q \wedge f$
		m			$i \vee e \vee \neg y$
		n			$l \vee \neg q \vee \neg i$
$\neg o$	o	o			$\neg O \wedge \neg Q$
$\neg p$	p	$\neg p$			$L \vee \neg W$
	q	q			$H \vee E$
r	r	r			$\neg D \vee Z$
s	$\neg s$	s			$\neg Y \wedge X$
$\neg t$		t			$\neg W \wedge \neg L$
u	u	$\neg u$	$\neg D \vee F$		
v	v	$\neg v$	$J \vee \neg P$		
$\neg w$	$\neg w$	$\neg w$	$Q \wedge N$		
x	x	x	$\neg O \vee R$		
y		y	$\neg W \wedge \neg P$		
		z		$F \vee \neg J$	
$\neg D$		$\neg D$		$L \vee M$	
$\neg E$		E		$P \vee W$	
F	F			$M \wedge \neg P$	
		$\neg H$		$\neg S \vee U$	
J	$\neg J$				
	$\neg L$	$\neg L$			
	M				
N					
$\neg O$		$\neg O$			
$\neg P$	$\neg P$				
Q		$\neg Q$			
R					
	$\neg S$				
	$\neg U$				
	V				
$\neg W$	$\neg W$	$\neg W$			
		X			
		$\neg Y$			
		Z			
$\neg(\neg d \wedge \neg z)$	$\neg(\neg d \wedge \neg z)$				
$\neg w \wedge s$	$\neg(\neg w \wedge s)$				
$a \vee \neg u$					
$h \wedge i$	$h \wedge i$				
$\neg b \vee \neg w$					
$t \wedge \neg c$					
$\neg s \vee \neg f$	$\neg s \vee \neg f$				
$\neg c \vee t \vee o$	$\neg c \vee t \vee o$				
$\neg z \vee f \vee o$	$\neg(\neg z \vee f \vee o)$				
$p \wedge i \wedge \neg b$					

Table A.1: Initial Facts in Three Knowledge Bases

Proposition	State	Proposition	State
a	u	$\neg d \wedge \neg z$	f
b	u	$\neg w \wedge s$	u
c	u	$a \vee \neg u$	u
d	t	$h \wedge i$	u
e	u	$\neg b \vee \neg w$	u
f	u	$t \wedge \neg c$	u
g	f	$\neg s \vee \neg f$	u
h	u	$\neg c \vee t \vee o$	u
i	u	$\neg z \vee f \vee o$	u
j	f	$p \wedge i \wedge \neg b$	u
k	k	$m \vee \neg b$	u
l	u	$n \wedge t$	u
m	u	$k \vee wveep$	t
n	u	$\neg x \wedge l \wedge e$	u
o	u	$\neg b \vee \neg w$	u
p	u	$\neg q \wedge \neg m$	u
q	u	$\neg e \vee \neg m$	u
r	r	$z \wedge \neg i$	u
s	u	$\neg s \vee c$	u
t	u	$p \vee i$	u
u	u	$t \wedge \neg c \wedge b$	u
v	u	$n \wedge q \wedge f$	u
w	u	$i \vee e \vee \neg y$	u
x	u	$l \vee \neg q \vee \neg i$	u
y	u	$\neg O \wedge \neg Q$	u
z	u	$L \vee \neg W$	t
D	f	$H \vee E$	u
E	u	$\neg D \vee Z$	t
F	t	$\neg Y \wedge X$	t
H	u	$\neg W \wedge \neg L$	u
J	u	$\neg D \vee F$	t
L	u	$J \vee \neg P$	u
M	u	$Q \wedge N$	u
N	u	$\neg O \vee R$	t
O	u	$\neg W \wedge \neg P$	u
P	u	$F \vee \neg J$	t
Q	u	$L \vee M$	u
R	t	$P \vee W$	u
S	f	$M \wedge \neg P$	u
U	f	$\neg S \vee U$	t
V	t		
W	u		
X	t		
Y	f		
Z	t		

where t represents "*true*", f represents "*false*", u represents "*unknown*"

Table A.2: The Final Result After Merging Three Knowledge Bases