Revision in ACMS

by

Alex Kean

Technical Report 91-1
April, 1991

e-mail: kean @cs.ubc.ca

Department of Computer Science
University of British Columbia
Vancouver, B.C.
CANADA V6T 1W5

Revision in ACMS

Alex Kean

Department of Computer Science, University of British Columbia,
Vancouver, British Columbia V6T 1W5, Canada

Technical Report 91-1

April 1991
Email: kean@cs.ubc.ca

The motivation for creating truth maintenance systems is two fold. First, it is used for the abduction process of generating explanation; and second, to perform the necessary *bookkeeping* of revision of the knowledge base. The process of revision is defined as addition and deletion of knowledge from the knowledge base. A logical scheme for tracking conclusions in an assumption based clause management system (ACMS) for the purpose of abduction and revision is proposed. As a consequence, an incremental deletion scheme is derived. A protocol for assumption revision is demonstrated by a backtrack search example. The proposed ACMS is the first truth maintenance system that employs incremental deletion as part of its capability.

1 Introduction

Truth maintenance was a concept developed in Artificial Intelligence specifically to deal with problems of maintaining and querying knowledge that may vary over time. A truth maintenance system is usually a domain independent sub-system performing the task of knowledge management for a more domain dependent *Reasoner*. To date, there have been a number of influential developments namely, the *fact garbage collection* of Stallman and Sussman [12], Doyle's *justification-based truth maintenance* [2], de Kleer's *assumption-based truth maintenance* [1]

and *clause management* [10, 13, 5] [1].

The functionality of truth maintenance is two fold. Firstly it serves as an *abductive* inference engine for finding *explanations* for queries with respect to its knowledge base Σ at some instant in time. To be precise, an *explanation* E for a query G with respect to the knowledge base Σ is defined as

Definition 1.1

1. $\Sigma \models E \rightarrow G$
2. $\Sigma \cup E$ *is consistent.*

For instance, if its present knowledge base Σ contains $\{a \rightarrow b, b \rightarrow c\}$, then some of the *explanations* for the query c are a and b. We shall call this functionality abductive inference in truth maintenance. Most of the truth maintenance systems mentioned above handle abductive inference. A complete development of this feature, sometimes known as abductive reasoning, is reported in Kean and Tsiknis *Assumption-based Clause Management Systems* (ACMS) [5].

The second functionality we will appropriately call revision in truth maintenance. Revision is classified as two operational concepts, namely *addition* which means adding new knowledge into Σ, and *deletion* which means deleting existing knowledge from Σ. Revision is a concept which came about to capture the dynamic and non-monotonic nature of reasoning, that is to handle changes in knowledge over time. Hidden within revision is the ability to manage potentially conflicting *assumptions* used during the course of abductive inference. An *assumption*, is a statement asserted that may be retracted later, that is an *assumption* is a defeasible statement.

For instance, if the knowledge base is $\Sigma = \{a \rightarrow q, b \rightarrow r\}$ and the current set of assumptions is $\mathcal{A} = \{p \rightarrow a, \neg p \rightarrow b, \neg b\}$, then one explanation using only assumptions for the query q with respect to Σ is

$$\neg b \wedge (p \rightarrow a) \wedge (\neg p \rightarrow b).$$

[1] A good review on these systems except clause management can be found in Ramsay's book [8, pp 212–231]

Deductively, using the assumptions $\neg b$ and $\neg p \rightarrow b$ yields the consequence p. Furthermore, using the other assumption $p \rightarrow a$ yields the consequence a and with the fact $a \rightarrow q$, we can conclude the query q. Thus, if I were to retract one of the assumption, let say $\neg p \rightarrow b$, then the above chain of reasoning no longer holds. The same problem arises when a new assumption is added, where in this case new conclusions can be drawn that were not concluded before. Note that in terms of finding explanations, both the deletion of an assumption and addition of fact or assumption, may result in either generating more explanation or finding less explanations than before. This is due to the consistency requirement on explanations in definition 1.1.

Hereafter, I shall refer to the tasks of truth maintenance in performing *abductive inference* and *revision* as *clause management*. Earlier work on *abductive inference* and *addition* in clause management are reported in [10, 13, 5].

To have a *clause management system* that performs the above functions, some adequacy issues must be addressed. First, a naive but simple and correct approach would be to deduce explanation on-the-fly everytime we need to explain a query. We shall call this the interpreted approach. One example of such an approach is David Poole's THEORIST [7]. Thus, revising our assumptions over time involves simply deleting and adding assumptions in the set of assumption ($\mathcal{A}$). One disadvantage of such approach is that there may be many conclusions drawn for one query which remain valid for the next query. It would be ideal if we could "*remember*" conclusions drawn earlier and reuse them whenever possible making query processing more efficient. Thus, one requirement of *clause management* is that it must be able to save and reuse conclusions. We shall call this approach the compiled approach. All truth maintenance systems mentioned earlier use the compiled approach. A logical characterization of compilation in terms of *implicates/implicants* can be found in [10, 13, 3].

The second adequacy is the issue of *revision*. Since we are saving conclusions (compiling), every addition of knowledge or assumptions to the existing knowledge base requires re-compiling, hopefully only those relevant and affected conclusions. This is called the *update-problem* in clause management by Reiter and de Kleer [10] and one solution to this is an incremental addition algorithm reported in [4].

In duality, deleting an assumption requires the removal of those saved conclusions which depend on this assumption. A naive approach is simply by

removing all the saved conclusions and performing the re-compilation without the deleted assumption. This is not desirable since many conclusions may remain valid independent of the deleted assumption. Thus, an **incremental deletion** scheme is needed. All existing truth maintenance systems do not consider incremental deletion (or even deletion) as part of its functionality.

For instance, in Doyle's JTMS, a proposition P is either IN or OUT indicating whether P is consistent to be believed or not. Thus, believing P is OUT is not the same as asserting P is false. Similarly, deleting P (or P is not there) is neither false nor being OUT. Doyle does not consider the later property of deleted assumption in his JTMS [2]. In de Kleer's ATMS, a separate knowledge base is used to keep inconsistent assumptions (or *nogoods*). Thus, revising an assumption A is to assert $A \rightarrow false$ (or A is a *nogood*). Unfortunately, if we change our mind about A being *nogood* to *good* later, there is no way to revise except to delete A from the *nogood* knowledge base. Also, if the revised assumption occurs in many other *nogoods*, it is not clear how the deletion process can be achieved. de Kleer does not consider deletion as a function in his ATMS [1].

This paper will present a logical scheme of incremental deletion based on the framework of the *Assumption based Clause Management System* (ACMS) in [5]. The proposed ACMS is the first truth maintenance system that includes incremental deletion as part of its functionality.

In section 2, an ACMS is defined for the purpose of performing the tasks of clause management satisfying the adequacies stated. In section 3, a protocol for using the ACMS in performing intelligent backtracking is proposed and an example demonstrating the protocol is presented. In section 4, an argument is presented for the question *"is deletion necessary ?"* thus justifying the inclusion of incremental deletion as part of the functionality of clause management. Finally, conclusions and future works can be found in section 5

2 An ACMS

In this section we define an *Assumption Based Clause Management System* (ACMS) that is capable of performing assumption based abductive reasoning and revision. We restrict the set of facts $\mathcal{F}$ and the set of assumptions $\mathcal{A}$ of the

assumption based theory $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ to be propositional and finite. Without lost of generality we shall also assume that $\mathcal{F}$ and $\mathcal{A}$ are sets of sentences in CNF.

2.1 Abductive Reasoning

The abductive inference performed by the ACMS including assumptions is defined as follows:

Definition 2.1 *Let facts $\mathcal{F}$ and assumptions $\mathcal{A}$ be sets of sentences; and let the explanation E and query G be sentences, all in CNF.*

1. $E \subseteq \mathcal{A}$,
2. $\mathcal{F} \models E \rightarrow G$ *and*
3. $\mathcal{F} \cup E$ *is consistent.*

The explanation E is a minimal explanation for G with respect to $\mathcal{F}$ if no $E' \subset E$ is an explanation for G with respect to $\mathcal{F}$.

Intuitively, an explanation E for a query G with respect to $\mathcal{F}$ is a subset of assumptions, in conjunction with $\mathcal{F}$, is consistent and sanctions the query G. For instance, if the set of assumptions is $\mathcal{A} = \{a, b, c, d\}$ and the set of facts is $\mathcal{F} = \{a \rightarrow b, b \rightarrow c, c \rightarrow d\}$, then the explanation a is an assumption we would use to explain the query c.

2.2 Compilation

Ignoring the set of assumptions for a moment, the method used to compile the set of facts $\mathcal{F}$ such that we could re-use the conclusions repeatedly, is to transform them into a set of equivalent sentences of *minimal implicates*.

Definition 2.2 *Let $\mathcal{F}$ be a set of sentences and P a sentence in CNF.*

1. *P is an implicate of $\mathcal{F}$ if $\mathcal{F} \models P$.*

2. *P is a minimal implicate of $\mathcal{F}$ if there is no implicate P' of $\mathcal{F}$ such that $P' \subset P$.*

Thus, the set of all minimal implicates of $\mathcal{F}$, denoted by $MI(\mathcal{F})$ is a subset of $Th(\mathcal{F})$ and additionally possesses the logical property that $\mathcal{F} \equiv MI(\mathcal{F})$ [13]. Using the same example as before, let $\mathcal{F} = \{a \rightarrow b, b \rightarrow c, c \rightarrow d\}$, the set of minimal implicates of $\mathcal{F}$ is $MI(\mathcal{F}) = \{a \rightarrow b, b \rightarrow c, c \rightarrow d, a \rightarrow c, a \rightarrow d, b \rightarrow d, a \rightarrow a, b \rightarrow b, c \rightarrow c, d \rightarrow d\}$. The reader immediately notice the compilation into minimal implicates is almost like computing all the possible consequences from $\mathcal{F}$, except that only the minimal consequences are kept.

Consequently, an explanation can be computed relatively easily from the set $MI(\mathcal{F})$ [13]. For instance, in a crude sense, finding an explanation for the query c means simply selecting the minimal implicates that have c in them and their antecendents will be the explanations. In the above example, $\{a, b, c\}$ are all minimal explanations for c with respect to $\mathcal{F}$.

2.3 α-theory

Let us return to the orginal theory $\mathcal{T}$ with assumptions. Ideally, for each assumption, we would like to pre-compute the set of conclusions it can derive together with $\mathcal{F}$. Unfortunately, we cannot simply union the set of assumptions $\mathcal{A}$ with $\mathcal{F}$ and compute the set of minimal implicates because $\mathcal{A}$ might contain contradicting assumptions. Thus, one approach is for each query G, to check whether any assumption is applicable to deduce G. This is unattractive since it does not excercise the principle of "remembering" conclusions and it does not utilize the compilation of $\mathcal{F}$.

How do we pre-compute and "keep track" of conclusions deduced using assumptions which are potentially contradictory ? We shall call this the *tracking problem in the compiled approach.* All existing truth maintenance systems resolve this problem by keeping pointers, in the computer programming sense, to track the conclusions, as well as elaborate data-structures to handle the existance of contradicting assumptions. For instance in Doyle's *justification based truth maintenance system*, an explicit data-structure of IN and OUT labels denoting

the status of proposition is used, and methods to resolve the status being IN or OUT are algorithmic [2].

We shall now present a logical scheme for the tracking problem. It is well known in mathematical logic that *conservative extension by explicit definition* preserves the logical consequences of the old theory in an extended new theory [11, pp 57]. We apply this idea as follows: for each assumption $\alpha \in \mathcal{A}$, we introduce an explicit definition by defining $\alpha \equiv \lambda$, where λ is a new variable not occuring elsewhere in the theory. Subsequently, we can extend the theory $\mathcal{F}$ by adding all the explicit definitions introduced. More precisely,

Definition 2.3 (σ-transformation) *Let $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ be an α-theory . We define a transformation σ as follows.*

1. *For every sentence α in $\mathcal{A}$, $\sigma(\alpha) = \lambda$ where λ is a new propositional variable not used anywhere in the theory.*
2. $\sigma(\mathcal{A}) = \{\sigma(\alpha) \mid \alpha \in \mathcal{A}\}$
3. $\sigma(\mathcal{F}) = \mathcal{F} \cup \{\sigma(\alpha) \equiv \alpha \mid \alpha \in \mathcal{A}\}$ *and*
4. $\sigma(\mathcal{T}) = (\sigma(\mathcal{F}), \sigma(\mathcal{A}))$.

In the computer programming sense, the σ-transformation is *indexing* every sentence α in the assumption set $\mathcal{A}$ with a new variable λ. Thus, any new consequence derived from $\sigma(\mathcal{F})$ dependent on the assumptions will contain corresponding variables λ. Note that the sentence α might be non-atomic, in which case the interaction between the equivalent label λ and the set of facts $\mathcal{F}$ is not intuitive. If σ-transformation is performed on both $\mathcal{F}$ and $\mathcal{A}$, then $\mathcal{T}$ and $\sigma(\mathcal{T})$ are equivalent as expressed in the following theorem.

Theorem 2.1 *For any α-theory $\mathcal{T} = (\mathcal{F}, \mathcal{A})$, $\mathcal{T}$ and $\sigma(\mathcal{T})$ are equivalent in the sense that for any sentence G, E is an explanation of G from $\mathcal{T}$ iff $\sigma(E)$ is an explanation of G from $\sigma(\mathcal{T})$.*

Proof : Trivially follows from definition 2.3 and propositional reasoning. □

Using the σ-transformation, we can now safely compile the set $\sigma(\mathcal{F})$ into its corresponding set of minimal implicates. For example, let $\mathcal{A} = \{p, \neg p\}$ and $\mathcal{F} = \{p \rightarrow q\}$. Using σ-transformation we introduce two new variables λ_1 and λ_2 as new names for the assumptions. Thus $\sigma(\mathcal{F}) = \mathcal{F} \cup \{p \equiv \lambda_1, \neg p \equiv \lambda_2\}$ and its corresponding set of minimal implicates is

$$
\begin{array}{lll}
MI(\sigma(\mathcal{F})) = \{ & \neg(\lambda_1 \wedge \lambda_2), & \lambda_1 \vee \lambda_2, \\
& \lambda_2 \rightarrow \neg p, & \neg p \rightarrow \lambda_2, \\
& p \rightarrow \lambda_1, & \lambda_1 \rightarrow p, \\
& p \rightarrow q, & \\
& \neg\lambda_2 \rightarrow q, & \lambda_1 \rightarrow q\}.
\end{array}
$$

If the query is q, there is a minimal explanation, namely λ_1, denoting the fact that assuming $\lambda_1 \equiv p$ in $\mathcal{F}$ explains the conclusion q. If we accept the negation of assumptions as our explanation, that is $E \subseteq \mathcal{A} \cup \overline{\mathcal{A}}$ in definition 2.1, then $\neg\lambda_2$ is also a minimal explanation for q denoting the fact $\neg(\neg p)$.

2.4 Addition

In the process of addition, there are two types of knowledge namely new facts and new assumptions. In the case of adding a new facts C, we can simply compute the new set $MI(\ MI(\sigma(\mathcal{F})) \cup \{C\}\ )$. In the case of adding a new assumption γ,

1. $\sigma(\mathcal{A}) = \sigma(\mathcal{A}) \cup \{\lambda\}$ for a new variable λ not used anywhere;
2. compute the set $MI(\ MI(\sigma(\mathcal{F})) \cup \{\gamma \equiv \lambda\}\ )$.

In both cases, an incremental algorithm to update the set of minimal implicates can be used [4]. Note that the addition of facts or assumptions into the theory $\mathcal{T}$ might produce inconsistency. The proper procedure for adding facts or assumptions is an issue in the usage of the ACMS. A protocol for this usage is proposed in section 3.

2.5 Deletion

In terms of deletion, only assumptions are removable. The distinction between facts and assumptions is not clear. One would argue that there is no facts but assumptions since knowledge changes over time. Consequently, everything is removable. Conversely, a reasonable definition for facts is the knowlegde that remains unchange during the course of the problem solving. In this paper, we shall adopt the later position for simplicity in the presentation.

When an assumption α is chosen for deletion, all consequences in this case all the minimal implicates, derived involving α must be removed. How would we indentify these consequences ? Recall that each assumption has a unique variable λ attached to it as an equivalence. Thus, any consequence P that can be derived using α, but is not derivable without it, must have λ occuring in P. This is expressed by the following theorem, which is a form of conservative extension by explicit definition [11, pp 57].

Theorem 2.2 *Let $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ be an α-theory , α be a sentence and λ be a new propositional variable not occurring in $(\mathcal{F}, \mathcal{A})$.*

$$\mathcal{F} \not\models P \text{ and } \mathcal{F} \cup \{\alpha \equiv \lambda\} \models P \text{ only if } \pm\lambda \text{ occurs in } P.$$

Proof : Assume that $\pm\lambda$ does not occur in P. Let $\mathcal{F} \not\models P$, then there exists a model $M \models \mathcal{F}$, $M \not\models P$. Since λ is a distinct variable not occurring in $(\mathcal{F}, \mathcal{A})$, the extended model $M \cup \{\lambda\} \models \mathcal{F}$ where λ assumes the truth value of α and $M \cup \{\lambda\} \not\models P$ because $\pm\lambda$ does not occur in P. But the extended model $M \cup \{\lambda\} \models \mathcal{F} \cup \{\alpha \equiv \lambda\}$ and by the fact that $\mathcal{F} \cup \{\alpha \equiv \lambda\} \models P$. The same must be true for the extended model, $M \cup \{\lambda\} \models P$, contradicting the assumption. □

As a consequence, deleting an assumption $\alpha \equiv \lambda$ in $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ is achieved by deleting all the conclusions P that have $\pm\lambda$ occurring in them. In the case of $\sigma(\mathcal{T})$, the deletion consists of simply the set operation

1. $\sigma(\mathcal{A}) = \sigma(\mathcal{A}) - \{\lambda\}$ and
2. $\sigma(\mathcal{F}) = \sigma(\mathcal{F}) - \{\alpha \equiv \lambda\}$.

Interestingly enough, since the set of all conclusions of the α-theory is represented by $MI(\sigma(\mathcal{F}))$ that is, the set of all minimal conclusions, deleting an assumption $\alpha \equiv \lambda$ in $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ is defined in the following way.

Definition 2.4 (λ-Deletion) *Let $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ be an α-theory , $\sigma(\alpha) = \lambda$ for $\alpha \in \mathcal{A}$, $\lambda \in \sigma(\mathcal{A})$ and let $MI(\sigma(\mathcal{F}))$ be the set of all minimal implicates of $\sigma(\mathcal{F})$. The revised theory $\sigma(\mathcal{T})' = (MI(\sigma(\mathcal{F}))', \sigma(\mathcal{A})')$ is defined as*

1. $\sigma(\mathcal{A})' = \sigma(\mathcal{A}) - \{\lambda\}$
2. $MI(\sigma(\mathcal{F}))' = MI(\sigma(\mathcal{F})) - \{P \mid P \in MI(\sigma(\mathcal{F}))$ *and* $\pm\lambda$ *occurs in* $P\}$.

The correctness of the above λ-deletion is rather trivial since $\sigma(\mathcal{F})$ is equivalent to $MI(\sigma(\mathcal{F}))$ in the sense that $\sigma(\mathcal{F}) \models P$ if and only if there is a $P' \in MI(\sigma(\mathcal{F}))$ that subsumes P [13]. Additionally, the λ-deletion process involves only a linear search through the set $MI(\sigma(\mathcal{F}))$. With careful indexing on the set $MI(\sigma(\mathcal{F}))$ a faster method is feasible.

3 A Protocol

In this section, a protocol is proposed for performing assumption revision while using the ACMS in performing intelligent backtracking search. The protocol proposed here is of the same nature as Frank Ramsey's test for evaluating conditionals suggested in 1929, summed up by Stalnaker [14, pp 95] as follows :

> [Ramsey's Test] *This is how to evaluate a conditional: first, add the antecedent (hypothetically) to your stock of beliefs; second,* **make** *whatever adjustments are required to maintain consistency (without modifying the hypothetical belief in the antecedent); finally, consider whether or not the consequent is then true.*

The key idea lies in the statement "... make whatever adjustments are required to maintain consistency ...". In our framework of assumption based reasoning, the Ramsey's Test takes the form of asserting an assumption in the

theory $\mathcal{T} = (\mathcal{F}, \mathcal{A})$ such that the set of assumptions $\mathcal{A}$ asserted so far is consistent with $\mathcal{F}$. Thus, the actual sequence of asserting an assumption is charaterized as follows:

> [Asserting Assumption] *This is how to assert an assumption α into your current set of consistent assumptions $\mathcal{A}$: first, check if any of the minimal explanations E for α is inconsistent with your current set of assumptions $\mathcal{A}$; if it is not inconsistent add α into $\mathcal{A}$; otherwise delete the assumption E in favour of another alternate assumption other than E. Repeat the process until it is consistent to add α.*

A protocol comprised of $add(A)$, $delete(A)$ and $explain(A)$ is sufficient to simulate the above sequence of operations. To illustrate the above reasoning using this protocol, we shall mimic the reasoning of intelligent backtracking search in a constraint satisfaction problem [2].

Assuming the *Reasoner* is solving a constraint satisfaction problem using the domain independent ACMS. The actual algorithm for the *Reasoner* is based on the *operation of asserting assumptions* but the interpretation of the explanations is domain dependent to the *Reasoner*. The following example assumes the *Reasoner* is capable of interpreting the explanations as results.

Example 3.1 *Assume that we have three variables X, Y and Z, each variable can have a value of r (red) or w (white) and the constraints are $X \neq Y$ and $Y \neq Z$.*

Figure 1 shows the complete search space for the problem of finding consistent values for the variables X, Y and Z satisfying the constraints $X \neq Y$ and $Y \neq Z$. The edges of the search tree is labeled with the assignment of value for the variables and the constraints. In a naive top-down, left-to-right search strategy, the left-most branch $(X = r, Y = r, Z = r)$ is tried and upon failing at $X \neq Y$ (denoted by a cross $\times$), it backtracks to the most recent point, that is $Z = w$ and fails again at $X \neq Y$.

Note that an intelligent search strategy should immediately detect the cause of failure comes from the incompatible values bewteen X and Y. Thus,

[2] The demonstration of the protocol is not suggested as an efficient method to solve constraint satisfaction problems. An in depth study of constraint satisfaction problems can be found in [6]

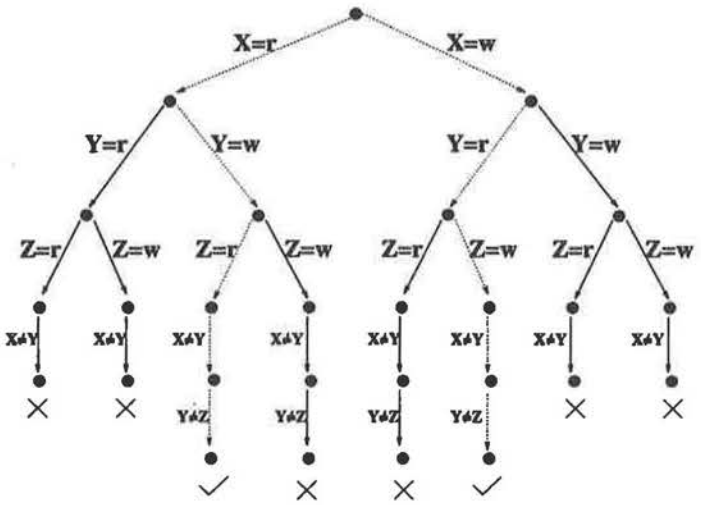

Figure 1: Complete search tree.

backtracking should occur at the point of X or Y, avoiding unnecessary trial of $Z = w$.

Using the protocol suggested, the following sequence of reasoning steps mimics the intelligent backtracking strategy mentioned above. Initially as facts, the uniqueness and existence for each value are expressed.

$$\begin{aligned}\mathcal{F} = \{ \quad & X = r \vee X = w, \quad \neg(X = r \wedge X = w), \\ & Y = r \vee Y = w, \quad \neg(Y = r \wedge Y = w), \\ & Z = r \vee Z = w, \quad \neg(Z = r \wedge Z = w) \quad \}.\end{aligned}$$

First, we shall make the postulate for each variable; $X = r, Y = r$ and $Z = r$ in our assumption set, that is [3]

$$\begin{aligned}\sigma(\mathcal{A}) = \{ & (X = r) \equiv \lambda_1, \\ & (Y = r) \equiv \lambda_2, \\ & (Z = r) \equiv \lambda_3 \quad \}.\end{aligned}$$

[3]Note that the assumption set $\sigma(\mathcal{A})$ contains only new variables λ_i. For clarity and ease of reference, the equivalence is written.

We shall assume that the set $\mathcal{F}$ and $\mathcal{A}$ are transformed into $\sigma(\mathcal{F})$ and $\sigma(\mathcal{A})$ by the σ-transformation. Also, the set $\sigma(\mathcal{F})$ is compiled into the set of minimal implicates $MI(\sigma(\mathcal{F}))$.

Considering the first constraint $X \neq Y$, we ask for the minimal explanation for it, that is $explain(X \neq Y)$, and we also assume the explanation is constraint over the set $\mathcal{A} \cup \overline{\mathcal{A}}$, and they are

$$\lambda_2 \wedge \neg\lambda_1 \text{ and } \lambda_1 \wedge \neg\lambda_2.$$

The first explanation says that if both statements $Y = r \wedge X \neq r$ are true then they sanction the constraint $X \neq Y$. Unfortunately $X \neq r$ contradicts with our earlier postulate $X = r$. Similarly, the second explanation expresses the alternate contradiction $X = r \wedge Y \neq r$. Arbitrarily, we shall try to resolve the first contradiction $X \neq r$.

An alternative for the assumption $X = r$ is $X = w$. Before we replace the old assumption by a new one, we have to ensure that the alternative is also consistent. This is achieved by asking for its minimal explanation that is, $explain(X = w)$, and the minimal explanation is $\neg\lambda_1$ or simply $X \neq r$, which is perfectly acceptable. Thus, we can safely delete the assumption $X = r$ ($delete(X = r)$) and replace it with $X = w$ ($add(X = w)$). Consequently, the current set of assumption becomes

$$\begin{aligned}\sigma(\mathcal{A}) = \{\!\{ & (X = w) \equiv \lambda_1, \\ & (Y = r) \equiv \lambda_2, \\ & (Z = r) \equiv \lambda_3 \ \}.\end{aligned}$$

Subsequently, we shall re-examine the constraint $X \neq Y$ with the new set of assumptions. By $explain(X \neq Y)$ we obtained as minimal explanations

$$\lambda_1 \wedge \lambda_2 \text{ and } \neg\lambda_1 \wedge \neg\lambda_2,$$

which express the fact that the constraint $X \neq Y$ is satisfied. Note that the explanation $\neg\lambda_1 \wedge \neg\lambda_2$ says that $X \neq w \wedge Y \neq r$ and by propositional reasoning, the specification of uniqueness and existence for variables imply $X = r \wedge Y = w$. In turn, we can $add(X \neq Y)$ into our assumption set and it becomes

$$\sigma(\mathcal{A}) = \{\!\{ (X = w) \equiv \lambda_1,$$

$$(Y = r) \equiv \lambda_2,$$
$$(Z = r) \equiv \lambda_3,$$
$$(X \neq Y) \equiv \lambda_4, \quad \}.$$

Considering the second constraint $Y \neq Z$, the minimal explanations for it are

$$\neg\lambda_2 \wedge \lambda_3,$$
$$\neg\lambda_3 \wedge \lambda_2,$$
$$\neg\lambda_1 \wedge \neg\lambda_3 \wedge \neg\lambda_4 \text{ and}$$
$$\lambda_1 \wedge \lambda_3 \wedge \neg\lambda_4.$$

The first two explanations basically express that differing values must be assign to variables Y and Z for the constraint $Y \neq Z$ to be satisfied. Again, there are two possible ways to resolve the conflict: either replace the value for Y, or for Z. Arbitrarily, lets resolve it by choosing an alternate value for Y that is, $Y = w$. Again, before replacing it we need to verify the consistency of this new asumption by $explain(Y = w)$ and the minimal explanations are

$$\neg\lambda_2 \quad \text{and} \quad \lambda_1 \wedge \neg\lambda_4.$$

Unfortunately the second explanation says that the conjunction of the assumptions $X = w$ and $\neg(X \neq Y)$ sanctions the choice $Y = w$. This is in direct conflict with our established constraint $X \neq Y$ and thus, the choice $Y = w$ is not acceptable.

This leaves us with only the variable Z, the alternative value is $Z = w$ and $explain(Z = w)$ yields the minimal explanation $\neg\lambda_3$ or simply $Z \neq r$. This is consistent since we are replacing $Z = r$ by $Z = w$. Hence, the new set of assumptions becomes

$$\sigma(\mathcal{A}) = \{(X = w) \equiv \lambda_1,$$
$$(Y = r) \equiv \lambda_2,$$
$$(Z = w) \equiv \lambda_3,$$
$$(X \neq Y) \equiv \lambda_4, \quad \}.$$

Subsequently, to re-examine the constraint $Y \neq Z$, the set of minimal explanations for $explain(Y \neq Z)$ are:

$\neg\lambda_2 \wedge \neg\lambda_3$,
$\lambda_2 \wedge \lambda_3$,
$\neg\lambda_1 \wedge \lambda_3 \wedge \neg\lambda_4$ and
$\lambda_1 \wedge \neg\lambda_3 \wedge \neg\lambda_4$.

and the first two explanations express the consistency of accepting $Y \neq Z$. Thus by adding it into the set of assumptions we obtain:

$$\begin{aligned}\sigma(\mathcal{A}) = \{\!\{&(X = w) \equiv \lambda_1,\\ &(Y = r) \equiv \lambda_2,\\ &(Z = w) \equiv \lambda_3,\\ &(X \neq Y) \equiv \lambda_4,\\ &(Y \neq Z) \equiv \lambda_5 \ \}\!\}.\end{aligned}$$

To verify the result, we compute $explain((X \neq Y) \wedge (Y \neq Z))$ to find the set of minimal explanations:

$\lambda_1 \wedge \lambda_2 \wedge \lambda_3$,
$\neg\lambda_1 \wedge \neg\lambda_2 \wedge \neg\lambda_3$,
$\lambda_1 \wedge \lambda_2 \wedge \lambda_5$,
$\lambda_4 \wedge \lambda_5$,
$\lambda_2 \wedge \lambda_3 \wedge \lambda_4$,
$\neg\lambda_2 \wedge \neg\lambda_3 \wedge \lambda_4$ and
$\neg\lambda_1 \wedge \neg\lambda_2 \wedge \lambda_5$.

The first minimal explanation $\lambda_1 \wedge \lambda_2 \wedge \lambda_3$ or simply

$$(X = w) \wedge (Y = r) \wedge (Z = w)$$

is a set of consistent assumptions for sanctioning the constraints $(X \neq Y) \wedge (Y \neq Z)$. Similarly, the explanation $\neg\lambda_1 \wedge \neg\lambda_2 \wedge \neg\lambda_3$ or simply

$$(X = r) \wedge (Y = w) \wedge (Z = r)$$

is an alternate solution.

4 Is Deletion Necessary ?

The previous sections have been presented under the assumption that deletion of knowledge is necessarily a feature of reasoning. Is deletion necessary ? As demonstrated in the previous section, if revision of assumptions is the strategy of reasoning as when evaluating conditionals, like intelligent backtracking search, then deletion is necessary as the complementary process to addition.

Also, recall that one of the reasons for compilation is to achieve fast retrieval. take for instance, the knowledge *if it is rainning, the ground is wet* and *if the ground is wet, I wear my rubber overshoes*. As a practical rule, we would compile it by adding *if it is rainning, I wear my rubber overshoes*, thus bypassing the intermediate step of reasoning. We shall call this compilation by addition because a new rule is added.

Conversely, consider Reiter's infamous default rule : p/q, read as *if it is consistent to conclude p, then conclude q* [9]. More intuitively, *if it is consistent that I have neckties (p), I will wear a necktie to the restaurant (q)*. Since I am poor and I don't have neckties, I cannot conclude that I wear a necktie to the restaurant. On a later occasion, I can afford and own neckties, I can now conclude that I will wear a necktie to the restaurant. Now owning neckties, the question becomes whenever I go to a restaurant, do I repeatly trying to verify whether it is consistent I have neckties before I wear one ? Since the change of status is lasting (does not have to be forever), wouldn't it be better to compile our new knowledge by deleting the default rule ? Thus, this notion of compilation by deletion will allow me to wear a necktie to the restaurant without suffering from headache by constantly worrying about consistency.

Conversely, deletion does not necessarily mean loss of information. The removal of an assumption usually means either the assumption is irrelevant or is inconsistent with my perceived current state of the domain of discourse. If it were irrelevant we could always keep this irrelevant assumption in a separate knowledge base, lets call it "dustbin", for the sake of keeping information around. If it were inconsistent, some would argue that removal of it loses information as when human reasons with inconsistency. I argue that *reasoners do not reason* **with** *inconsistency, rather* **about** *inconsistency*. Thus the detection of inconsistency within subject matter is a piece of knowledge at a different knowledge level from the subject matter in the domain of discourse. In the *Reasoner-ACMS* paradigm,

when the *Reasoner* detects inconsistency with the aid of the ACMS, a separate knowledge base is established to keep this piece of information. For instance, the *Reasoner* discovers that the assumptions A_1 and A_2 are inconsistent when used together in the knowledge base Σ. Thus in a separate knowledge base Σ_I responsible for maintaining inconsistent knowledge about Σ, an encoding in the form

$$'A_1' \wedge' \Sigma' \rightarrow \neg' A_2'$$

where the quotes indicate names can be used. Additionally this separate knowledge base can also be maintained by the ACMS. Thus I argue that each knowledge base should be kept *sterile* such that reasoning within the knowledge base is *hygienic*.

In short, deletion should not be viewed as a loss, but rather, in conjunction with addition, as a process of replacement of old assumptions with new ones. Additionally, not to be taken as refuting the argument above, the $ACMS$ is also powerful enough to reason **about** inconsistency within a single knowledge base [5].

5 Conclusions

In summary, the tasks of clause management involving abductive reasoning and revision can now be completely charaterized and computed in the logical framework of the ACMS. The tracking of conclusions is accomplished by a logical scheme of indexing and the deletion process is achieved by deleting those indexed conclusions. A protocol comprised of $add(A), delete(A)$ and $explain(A)$ is proposed and its application is demonstrated by mimicking the reasoning of intelligent backtrack search. The ACMS proposed here is the first clause management system that takes into account deletion as one of its functions.

As for future work, the choice of assumptions for deletion is an important issue. In the absence of domain specific knowledge, the preference over a set of equally plausible assumptions can be bewildering. With the help of the ACMS, some method of investigating the preference, such as the *canonical ordering* of explanations in the CMS [13], would be fruitful. Finally, different types of *Reasoners* use the ACMS differently and the study of their interaction could suggest a better definition for a protocol for the *Reasoner*-ACMS paradigm.

Acknowledgement: This paper is the result of the discussion with Hugo Chan on the subject of modelling exception handling in inheritance hierarchy for the purpose of database design. The author is also grateful to George Tsiknis for his collaboration on the original ACMS, Craig Boutilier for the reference to the Ramsey's test, and Peter Apostoli, Jane Mulligan, Alan Mackworth, David Poole, Greg Grudic for their comments and criticism.

References

[1] Johan de Kleer. An Assumption-based TMS. *Artificial Intelligence*, 28, 1986.

[2] Jon Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979.

[3] Alex Kean. The Approximation of Implicates and Explanations. Technical Report 31, Department of Computer Science, University of British Columbia, 1990.

[4] Alex Kean and George Tsiknis. An Incremental Method for Generating Prime Implicants/Implicates. *Journal of Symbolic Computation*, 9:185–206, 1990.

[5] Alex Kean and George Tsiknis. Assumption based Reasoning and Clause Management Systems. Technical Report 9, Department of Computer Science, University of British Columbia, 1990.

[6] Alan K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.

[7] David Poole, Randy Goebel, and Romas Aleliunas. Theorist: A logical Reasoning System for Defaults and Diagnosis. CS Research Report 6, Department of Computer Science, University of Waterloo, 1986. Logic Programming and Artificial Intelligence Group.

[8] Allan Ramsay. *Formal Methods in Artificial Intelligence*. Cambridge University Press, 1988.

[9] Raymond Reiter. A Logic of Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[10] Raymond Reiter and Johan de Kleer. Foundations of Assumption-Based Truth Maintenance Systems: Preliminary Report. In *Proceeding of AAAI-87*, pages 183–188, Seatle, Washington, 1987.

[11] Joseph R. Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967.

[12] R. M. Stallman and G. J. Sussman. Forward Reasoning and Dependency Directed Backtracking. *Artificial Intelligence*, 9:135–196, 1977.

[13] George Tsiknis and Alex Kean. Clause Management Systems (CMS). Technical Report 21, Department of Computer Science, University of British Columbia, 1988.

[14] Frank Veltman. *Logics for Conditionals*. University of Amsterdam Press, 1985.