Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Software maintenance via default theory evolution: Preliminary report

Aditya K. Ghose
School of Computing and Information Technology
Griffith University
Nathan, Brisbane QLD 4111 Australia
aditya@cit.gu.edu.au

Randy Goebel
Department of Computing Science
University of Alberta, Edmonton
Alberta, Canada, T6G 2H1
goebel@cs.ualberta.ca

Abstract

This paper addresses the problem of automated support for software maintenance from the viewpoint of research in belief revision and default reasoning. We argue that formal specifications should be viewed as mutable specification theories, *which we treat as default theories in a variant of default logic called PJ-default logic [3]. We show that this approach to analyzing formal specifications provides benefits not only in terms of a more powerful representation language, but also because it permits us to define a specification revision framework which supports reuse-oriented software maintenance. Our approach views specification revision as a process of mapping between default theories and provides us the ability to explicitly record retracted specifications and take them into account in deciding the outcomes of the specification revision process, the ability to retain specifications which would otherwise be discarded in a semantically well-founded manner in anticipation of future reuse. We argue that our approach is a language-independent method of analyzing and revising specifications in any formal specification language.*

1 Introduction: Software maintenance

A simple method of modelling the dynamics of software systems is to view the process as a mapping between *states*, where each state consists of three entities:

- A *context*, representing the environment in which the software system is situated.
- A *specification*, representing an abstraction of the software system design and functionality. Software specifications, unlike contexts, are representable artifacts. Specifications may be represented in a formal language, or via a collection of informal tools, often involving graphical interfaces. This second category of specification techniques are more user-friendly by insulating the programmer or analyst from the need to use a mathematically demanding formal specification language. This advantage comes at a high cost, however. In addition to losing the ability to make precise yet abstract definitions, informal specifications also lose out on the benefits of early defect removal, coding guidance and derivation of test data enjoyed by formal specification techniques. In this paper, we shall focus only on formal specification techniques.
- A *implementation* representing the implemented software system.

Software maintenance tasks are usually of three kinds:

- *Corrective:* This is driven by validation failures, i.e., situations where the specification does not correctly reflect the context, or by verification failures, i.e., situations where the implementation does not adequately reflect the specification.
- *Perfective:* This form of maintenance is usually undertaken to improve the performance of the software system, leaving system functionality unchanged.
- *Adaptive:* Adaptive maintenance is driven by changes in the context. A *context shift* necessitates a revision of the specification, and correspondingly, a change in the implementation. Ideally, the change in the implementation is minimal, permitting maximum reuse of existing software components.

In this paper, we shall focus on the corrective and adaptive maintenance tasks. Within the set of corrective maintenance tasks, we are interested only in those driven by validation failures (which result in specifications which do not correctly or adequately reflect the context). The common feature of all of these tasks is a process of *specification revision*, i.e., a process of updating the specification to reflect changes in the environment in which the software system is situated. Our goal in this paper is to propose a *language-independent* framework for specification revision. In other words, we shall define a set of specification revision techniques which can apply to any formal specification language which includes notions of consistency and logical consequence. An interesting consequence of this exercise will be the observation that the logical framework that we develop for this purpose is also a viable formal specification language in its own right.

We shall use the notion of *maximizing re-use* of software components as the guiding principle in our effort to develop a framework for specification revision. We shall make the simplifying assumption that maximal re-use can be achieved during the maintenance process by making minimal change to the specification. While this alone does not guarantee maximal re-use, it is well-recognized that this is an important contributing factor, and our assumption is therefore a useful one to make.

2 A framework for specification revision

Our work is motivated by the following observations:

- Current approaches to formal specifications provide little or no account of what it means to minimally revise specifications in the light of new inputs from the context.
- Existing work does not address the question of *mutable specifications* (with the possible exception of [13]). Informally, mutable specifications are tentative, default or defeasible specifications. Mutable specifications correspond to *E-type programs* in Lehman's [12] classification of programs as E-type or S-type. Roughly, S-type programs are those which are correct with respect to specifications which remain invariant over time. E-type programs, on the other hand, are those which are correct with respect to specifications which may, in fact, vary over time. Mutable specifications are useful in a variety of ways:
  - They can be used to represent specifications which are known *a priori* to be assumptions that may be potentially violated.
  - They can be used to *complete* specifications via the application of a set of default specifications relevant to the domain. Consider for instance the domain of software systems for chemical reactor control. It is possible to develop a set of system-independent specifications for such control systems which would include, for instance, assumptions that the system receives sensor inputs from temperature, pressure and pH sensors. Such assumptions could however be violated in specific systems, such as ones in which pressure sensors are redundant. Default specifications can thus be used to obtain complete specifications by taking a potentially incomplete set of specifications generated by system developers and applying all of the relevant defaults.
  - They can be used to represent, and retain, in a semantically well-founded manner, specifications which become questionable as a result of specification revision. A major role for mutable specifications in the framework we shall develop in this paper will be in this capacity.
- Two kinds of change can take place during specification revision: new specifications may be added and existing specifications may be withdrawn or retracted. Specifications that are added can be represented in an obvious manner in a formal specification framework. Existing work, however, provides no account of how retracted specifications may be represented, given that retracted specifications are as important as newly added specifications in determining the outcome of a specification revision process (as an example later will show).
- A guiding principle in re-use oriented software maintenance is the need to maintain repositories of software components not currently in use in anticipation of future re-use. Similar considerations apply to specifications. Specifications that become invalid as a result of specification revision should be retained in a semantically meaningful way in anticipation of future use. Current frameworks for formal specifications provide no support for this.

Our approach to specification revision will draw on results from the closely related areas of belief revision and default reasoning. We shall establish this connection in two stages. First, we shall argue that results from the area of belief revision provide a sound semantic notion of minimal change in specification revision. Second, we shall argue that specifications need to be viewed as default theories in order to account for mutable specifications, and in order provide an adequate account of re-use oriented specification revision.

Alchourrón, Gärdenfors and Makinson have undertaken a systematic study of the dynamics of belief change, resulting in what is currently popularly known as the AGM framework for belief change [1], [5], [?], [4]. In the AGM framework, the belief state of an agent is represented by a deductively closed, logically consistent set of propositional sentences called a *belief set*. They define three kinds of belief change operations: *expansion*, in which the new belief being added is guaranteed to be consistent with the existing body of beliefs; *contraction*, in which an existing belief is retracted; and *revision*, in which a new belief, which may possibly be inconsistent with existing beliefs, is added. The operations of contraction and revision can be defined in terms of each other, as shown by the *Levi identity* below (here, K^*_A,K^-_A and K^+_A denote, respectively, the revision, contraction and expansion of K with A): $K^*_A = (K^-_{\neg A})^+_A$. The *Harper identity* [11] ($K^-_A = K^*_{\neg A} \cap K$) similarly defines contraction in terms of revision. We decsribe only contraction operators in this section, since the corresponding revision operators follow via the Levi identity. Alchourrón, Gärdenfors and Makinson define a set of rationality postulates for each of the operations of expansion, revision and contraction. The postulates for contraction are listed below.

1- For any sentence A and any belief set K, K^-_A is a belief set.

2- $K^-_A \subseteq K$.

3- If $A \notin K$, then $K^-_A = K$.

4- If $\not\models A$, then $A \notin K^-_A$.

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

6- If $\models A \leftrightarrow B$, then $K^-_A = K^-_B$.

7- $K_A^- \cap K_B^- \subseteq K_{A \wedge B}^-$.

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

Postulate (1-) requires that beliefs be represented in the same form before and after a belief change step. (2-) requires that no new beliefs be held as a result of a contraction. (3-) requires that if the belief to be contracted is not held, then no change should be made. (4-) requires that every contraction operation succeed, unless the belief being contracted is a logical truth. (5-) is the priniciple of recovery, which requires that if a belief held in a given belief state is retracted and then added back to the belief state, the outcome contains the initial belief state, i.e., the initial belief state is recovered. (6-) is the priniciple of irrelevance of syntax, which requires that the outcome of a contraction operation be independent of the syntactic form of the beliefs being contracted. (7-) requires that the retraction of a conjunction of beliefs should not retire any beliefs that are common to the retraction of the same belief set with each individual conjunct. (8-) requires that, when retracting the conjunct of two beliefs A and B forces us to give up A, then in retracting A, we do not give up any more than in retracting the conjunction of A and B.

Our work on defining a semantically well-founded approach to specification revision will take this formalization of minimal change as its starting point. We shall therefore view specification revision as a process of minimally changing specification theories (note that we shall use the term *revision* to denote the general operation of theory change, as opposed to the specific form used in the AGM framework). However, we shall not treat specification theories as classical theories, in a significant departure from the AGM framework, as we shall see later in this section. We shall use a simpler repertoire of belief change operations for specification revision. Given that *expansion* is a purely theoretical artifact (one rarely has a guarantee in real-life situations that the new input will be consistent with the existing theory) we shall only consider the following two kinds of specification revision operations:

1. *Specification addition:* This involves adding a new specification to a specification theory, and admits the possibility that the new specification might contradict existing specifications in the specification theory.

2. *Specification retraction:* This involves forcing a specification theory to *not commit to* an assertion, and may involve withdrawing existing specifications from the specification theory.

Although our work shall take the AGM framework as its starting point, our approach shall involve a significant departure from the AGM account of belief change. Listed below are five reasons why the AGM approach to belief change cannot be directly applied for the purposes specification revision. We shall use these observations to motivate the need for an alternative framework for specification revision.

First, the AGM representation scheme uses classical propositional theories, which does not permit mutable specifications to be represented in a meaningful way.

Second, retracted assertions cannot be represented in the AGM framework. This results in a form of pathological behaviour illustrated in the following example.

Example: Consider the AGM framework. Let $\{b, f\}$ be the alphabet of our language. Let the initial theory be $K0 = Cn\{b \rightarrow f\}$. After contracting f, let the outcome be $K1 = Cn\{b \rightarrow f\}$ (since f is not a consequence of $K0$, no change is made to $K0$). Revising $K1$ with b results in the theory $Cn\{b, b \rightarrow f\}$. Thus, the assertion f returns to the theory, although the only new information (the assertion b) obtained since being told to retract f does not in itself require that f be held again. A more detailed analysis reveals that when $K1$ is revised with b, three different entities need to be considered:

A: $b \rightarrow f$ and its consequences hold.

B: f is retracted.

C: b holds.

Prioritizing these entities informally using a relation $>$, where $x > y$ denotes that x has higher priority over y, a variety of outcomes are possible.

- If $C > A > B$ then $K1_b^* = Cn\{b, b \rightarrow f\}$.
- If $A > C > B$ then $K1_b^* = Cn\{b, b \rightarrow f\}$.
- If $A > B > C$ then $K1_b^* = Cn\{b \rightarrow f\}$.
- If $B > A > C$ then $K1_b^* = Cn\{b \rightarrow f\}$.
- If $C > B > A$ then $K1_b^* = Cn\{b\}$.
- If $B > C > A$ then $K1_b^* = Cn\{b\}$.

Clearly the three distinct entities and their relative prioritization need to be considered in generating an outcome. □

We therefore need a language where retracted specifications can be explicitly recorded. Third, in the AGM framework, assertions which are contradicted by the new input are irretrievably discarded, i.e., the only way such assertions can be recovered is if they are provided as input to the specification revision process. Consider the situation where an assertion x ceases to hold on account of another input assertion y, which is subsequently retracted. Common-sense dictates that x should be held again since the reason it ceased to hold has been removed. In the context of a framework for specification revision which supports re-use, such behaviour is specially important. We need a framework where specifications are never irretrievably discarded, but can be recovered without having to explicitly assert the specification again.

Fourth, the AGM framework uses a choice function to select amongst multiple candidate outcomes of the revision/contraction process (we have ommitted details of these operators in our description of the AGM

framework for brevity). This choice function is not context-sensitive; it is determined entirely by the theory currently held, and in later accounts of iterated belief change, by the sequence of inputs as well [14]. If theory preference must be performed as part of the specification revision process, one would require the theory preference criteria to be context-sensitive since the revision process is driven entirely by the context (as opposed to the epistemic attitudes of an agent, as is often the case in situations covered by the theory of belief change). The AGM framework cannot therefore be directly applied to the problem of specification revision.

Finally, the AGM framework insists that every input to the belief change process be accepted (in the case of revision, the resulting theory must contain the input assertion while in the case of contraction, the resulting theory must necessarily not contain the assertion being retracted). In the case of specification revision, certain inputs may be deemed (by the context) to be of low priority relative to the other specifications in the specification theory, and may not be immediately accepted, or may be rejected. We therefore need a framework which does not automatically accept every input (a process sometimes referred to as *non-prioritized belief change* [10]).

We shall utilize an alternative approach, which uses a nonmonotonic logic instead of classical logic as the representation language, as the basis of a framework for specification revision. As a first step, we shall argue that specifications need to be represented as *mutable specification theories*. Formally, we shall define a mutable specification theory to be a set $\Delta = \{S_1, S_2, \ldots, S_n\}$ where each S_i is a classical theory referred to as a *specification theory* as defined earlier. We shall say that S_i is *supported by* Δ iff $S_i \in \Delta$. The intuition is that every specification theory represents a different set of mutable specifications being assumed to hold. We shall defined specification addition and retraction for mutable specification theories in the following manner:

- *Addition:* This involves a mapping to a new mutable specification theory such that every specification theory it supports includes the input specification as a consequence.
- Retraction: This involves a mapping to a new mutable specification theory such that every specification theory it supports does not include the specification being retracted.

Note that the above defines successful revision. In the framework we shall define, these operations will not always succeed, the outcome being determined by a choice function based on the priority accorded to the input specification relative to the existing ones. Formally, we shall treat mutable specification theories as *default theories*, specifically theories in a variant of Reiter's default logic [16] called PJ-default logic [3]. In PJ-default logic, default rules are restricted to be prerequisite-free and semi-normal (i.e., a PJ-default rule is of the form $\frac{:\beta}{\gamma}$ such that $\beta \models \gamma$). PJ-default logic improves over Reiter's default logic [16] by avoiding cases where Reiter's logic is too weak, preventing the derivation of "reasonable" conclusions (such as in the disjunctive default problem) as well as cases where Reiter's logic is too strong, permitting the derivation of unwanted conclusions (for a detailed discussion of these issues, see [3]). This approach has other useful properties as well, such as semi-monotonicity, the guaranteed existence of extensions, weak orthogonality of extensions and a constructive definition for extensions. PJ-default extensions are defined as follows:

Definition 1 [3] *Let (W,D) be a prerequisite-free semi-normal default theory. Define:*
$E_0 = (E_{J_0}, E_{T_0}) = (Cn(W), Cn(W))$
$E_{i+1} = (E_{J_{i+1}}, E_{T_{i+1}}) = (Cn(E_{J_i} \cup \{\beta \wedge \gamma\}), Cn(E_{T_i} \cup \{\beta\}))$
where

$$i \geq 0,$$
$$\frac{:(\beta \wedge \gamma)}{\beta} \in D,$$
$$\neg(\beta \wedge \gamma) \notin E_{J_i}.$$

Then E is a PJ-extension for (W,D) iff
$E = (E_J, E_T) = (\bigcup_{i=0}^{\infty} E_{J_i}, \bigcup_{i=0}^{\infty} E_{T_i})$.

In the rest of the paper, whenever we refer to an extension, we shall refer to the E_T part of a PJ-extension. We shall use $E(\Delta)$ to denote the set of extension of a PJ-default theory Δ. We shall identify the set of specification theories supported by a mutable specification theory with the set of extensions of the corresponding PJ-default theory.

Definition 2 *A semi-normal default theory (W, D) is said to be uniform if for any two default rules $\frac{\alpha_i : \beta_i \wedge \gamma_i}{\beta_i}, \frac{\alpha_j : \beta_j \wedge \gamma_j}{\beta_j} \in D, \gamma_i = \gamma_j$ (if $\mid D \mid = 1$, then the theory is trivially uniform).*

In the paper, we shall be interested only in a special class of PJ-default theories, specifically that of *uniform* PJ-default theories.

In the next section we shall specification revision procedures based on this framework.

3 Specification revision procedures

We shall base the specification revision procedures presented in this section on our earlier work on belief change operators in [7] (some of these results are in turn based on results in [8] and [9]), but we shall have to deviate from the original belief change operators in several significant ways.

Until now, we have focussed on mutable specifications. Practical situations usually also involve a set of *immutable specifications* which must necessarily hold in every context. In our approach we shall account for this by assuming a preprocessing stage in which every addition or retraction is tested against the set of immutable specifications for viability. If a specification addition operation involves a new specification which is inconsistent with the immutable specifications, then the input is rejected. If a specification retraction operation involves retracting an immutable specification, then too the input is rejected.

Formally we shall define a generic specification revision (addition or retraction) operator to be a function of the following kind:

$$R : MST \times BP \times TP \times OP \times L \rightarrow MST \times ST$$

where MST is the class of mutable specification theories, ST is the class of specification theories, $OP = \{addition, retraction\}$, L is the language and BP and TP are two special classes of choice functions to be defined below. Intuitively, a specification reivison takes as input a mutable specification theory, two choice funtions, a boolean indicator denoting whether the operation is an addition or a retraction, and a sentence to be added or retracted, and produces as output another mutable specification theory and distinguished specification theory supported by the new mutable specification theory which represents the new preferred set of specifications.

Let us first analyze the representation language of mutable specification theories in greater detail. Given a mutable specification theory $\Delta = (W, D)$, we can identify the following distinguished sets of specifications:

- *Accepted specifications:* These are the specifications contained in W. Necessarily, the set of accepted specifications is contained in every specification theory supported by the mutable specification theory.
- *Retracted specifications:* These are the assertions contained in γ, where $\frac{:\beta_i \wedge \gamma}{\beta} \in D$, treating γ to be the set of its conjuncts if γ were written in conjunctive normal form. Necessarily, the negations of the elements of the set of retracted specifications are not contained in any of the specification theories supported by the mutable specification theory.
- *Tentative specifications:* This is comprised of the consequents β of every default rule $\frac{:\beta_i \wedge \gamma}{\beta} \in D$. A tentative specification may or may not be contained in a specification theory supported by the mutable specification theory.

A successful specification addition will result in the new specification being included in the the set of accepted specifications of the resulting mutable specification theory. Similarly, a successful specification retraction operation will result in the negation of the retracted specification being included in the set of retracted specifications.

We shall define a *specification base* to be any set consisting of accepted and retracted specifications. We shall syntactically disntinguish accepted and retracted specifications in the following manner: accepted specifications will be written as literals of the form ϕ while retracted specifications will be written as $-\phi$ where ϕ is a literal; the $-$ symbol denotes that the negation of ϕ does not hold. Given the special status of retracted specifications as constraints used for testing consistency, as opposed to actual assertions, we cannot refer to the consistency of a specification base. We shall use a notion of *compatibility* instead. Straightforwardly, a specification base is *compatible* if the set obtained by treating each retracted specification $-\phi$ as the literal ϕ (other things being the same) is consistent. We shall define a $Maxcomp$ operator as follows:

$Maxcomp(b) = \{b' \mid b' \subseteq b, b'$ is compatible and for all b'' s.t. $b' \subset b'' \subseteq b$, b'' is incompatible$\}$

where b, b' and b'' are specification bases. The $Maxcomp$ operator thus identifies maximal compatible subsets of specification base, given a possibly incompatible specification base as input. Given a specification base b, $s_a(b)$ denotes the set of accepted specifications in b while $s_r(b)$ denotes the set of retracted specifications

Recall that the specification revision operator takes as input two choice functions. The first of these belongs to the class of base preference (or BP) functions. Formally a base preference choice function c_b is defined as follows:

$$c_b : 2^B \rightarrow B$$

where B denotes the class of possible specification bases (given a formal specification language). Thus c_b takes a set of specification bases and selects one. The second choice function provided as input to the specification revision function belongs to the class of theory preference (or TP) functions. Formally a theory preference choice function c_t is defined as follows:

$$c_t : 2^T \rightarrow T$$

where T denotes the class of possible theories (given a formal specification language). Thus c_t takes a set of theories and selects one.

We can now provide a precise definition of the specification revision function R. We assume that the initial mutable specification theory is given by the PJ-default theory (W, D) where both W and D are finite. We assume that the dummy default $\frac{:\top}{\top} \in D$ for any (W, D) representing a mutable specification theory (this permits us to record retracted specifications even when there are no tentative specifications. First we define $s_{initial}$, a function which generates the specification base consisting of the initial set of accepted and retracted specifications, given the initial mutable specification theory as follows:

$s_{initial}((W, D)) = W \cup \{\gamma \mid$ there exists some $\frac{:\beta_i \wedge \gamma}{\beta} \in D\}$

We then define a function s_{inter} which generates the intermediate specification base by adding to the initial specification base a new accepted specification (in the case of addition) or a retracted specification (in the case of retraction).

$$s_{inter}(op, b, in) = \begin{cases} b \cup \{in\} & \text{if } op = addition \\ b \cup \{-\neg in\} & \text{if } op = retraction \end{cases}$$

where op denotes the operation (addition or retraction), b denotes a specification base, and in denotes the input sentence (to be added or retracted).

Let $\Delta = (W, D)$ be the initial mutable specification theory. Let c_b and c_t be the base preference and theory preference choice functions provided as input. As before, let *op* denote the revision operation (addition or retraction) and *in* denote the input specification to be added or retracted. Let Δ' denote the resulting mutable specification theory. t denotes the preferred specification theory supported by Δ' after the revision step.

$$R(\Delta, c_b, c_t, op, in) = (\Delta', t)$$

where $\Delta' = (W', D')$, with
$W' = s_a(c_b(Maxcomp(s_{inter}(op, s_{initial}(\Delta), in))))$
$D' = \{\frac{:\delta_i \wedge (\bigwedge s_r(c_b(Maxcomp(s_{inter}(op, s_{initial}(\Delta), in)))))}{\delta_i} \mid \delta_i \in (W'' - W') \} \cup$
$\{\frac{:\beta_i \wedge (\bigwedge s_a(c_b(Maxcomp(s_{inter}(op, s_{initial}(\Delta), in)))))}{\beta_i} \mid \frac{:\beta_i \wedge \phi_i}{\beta_i} \in D \}$ where $W'' = W$ if $op = retraction$ and $W'' = W \cup \{in\}$ if $op = addition$.

$$t = c_t(E(\Delta'))$$

Here $\bigwedge b$ stands for the conjunction of all the elements of b.

For brevity, we shall not provide a detailed example of specification revision here. We shall, however, rework the example used in Section 3 to motivate the need for explicit representations for contracted assertions to show how our approach addresses this issue.

Example: Let the initial mutable specification theory be given by:

$$(W_1, D_1) = (\{b \rightarrow f\}, \{\frac{:\top}{\top}\})$$

The mutable specification theory obtained by contracting f is given by:

$$(W_2, D_2) = (\{b \rightarrow f\}, \{\frac{:\top \wedge \neg f}{\top}\})$$

The mutable specification theory obtained by further revising with b is given by:

$$(W_3, D_3) = (\{b\}, \{\frac{:(b \rightarrow f) \wedge \neg f}{(b \rightarrow f)}, \frac{:\top \wedge \neg f}{\top}\})$$
if $c_b(Maxcomp(s_{inter}(addition, s_{initial}((W_2, D_2)), b))) = \{b, \neg\neg f\}$

(W_3, D_3) has a single extension, given by $Cn(\{b\})$. Notice that the contraction of f persists since f is in no extension of (W_3, D_3). $b \rightarrow f$ is demoted in status from an accepted specification in (W_2, D_2) to a tentative specification in (W_3, D_3). Notice also we would get a different outcome:

$$(W_3', D_3') = (\{b \rightarrow f\}, \{\frac{:b \wedge \neg f}{b}, \frac{:\top \wedge \neg f}{\top}\})$$
if $c_b(Maxcomp(s_{inter}(addition, s_{initial}((W_2, D_2)), b))) = \{b \rightarrow f, \neg\neg f\}$

and yet another outcome:

$$(W_3'', D_3'') = (\{b, b \rightarrow f\}, \{\frac{:\top}{\top}\})$$
if $c_b(Maxcomp(s_{inter}(addition, s_{initial}((W_2, D_2)), b))) = \{b, b \rightarrow f\}$

We can thus get each of the three distinct outcomes discussed in our motivating example, depending on the outcome of the base preference choice function c_b. □

4 Properties

In Section 3, we presented a set of desiderata for specification revision systems. In this section, we shall establish that our approach does, in fact, satisfy these requirements. First, we shall examine how our approach measures up against the requirements for minimal change presented in the AGM framework. Second, we shall establish that specifications are always recoverable in our framework. Together, these properties will show that our approach provides a good basis for a reuse-oriented software maintenance.

Let us consider the AGM criteria for minimal change first. Our formalization cannot be evaluated using the AGM postulates directly, for the following two reasons. First, the AGM postulates consider transitions between belief states represented as a single deductively closed propositional theory. Our operator maps between collections of theories (the multiple possible extensions of the PJ-default theories). Second, since the AGM postulates consider belief change as a single step process, it is difficult to evaluate minimal change over iterated steps. It is possible, however, to articulate a reformulated version of these postulates, and show that our framework satisfies them under certain conditions. We shall formulate these conditions first. The following defines the class of *imperative* specification revisions.

Definition 3 *A specification revision operation* $R(\Delta, c_b, c_t, op, in)$ *is* imperative *iff:*

- $in \in c_b(Maxcomp(s_{inter}(op, s_{initial}(\Delta), in)))$ *if* $op = addition$.
- $\neg\neg in \in c_b(Maxcomp(s_{inter}(op, s_{initial}(\Delta), in)))$ *if* $op = retraction$.

The following result establishes that imperative operations always succeed.

Theorem 1 *If a specification revision operation* $R(\Delta, c_b, c_t, op, in) = (\Delta', t')$ *is* imperative*:*

- $\forall e : e \in E(\Delta')$, $e \models in$ *if* $op = addition$.
- $\forall e : e \in E(\Delta')$, $e \not\models in$ *if* $op = retraction$.

Since every belief change operation succeeds in the AGM, and since our specification revision operator avoids this in the general case, we can only establish connections between imperative specification revision operations and the AGM framework.

We shall also find it useful to establish the following technical result.

Lemma 1 *Let* $R(\Delta, c_b, c_t, op, in) = (\Delta', t')$. *Then* Δ' *is a uniform PJ-default theory if* Δ *is uniform.*

We need to further constrain the specification revision operation to make connections with the AGM framework. Specifically, we need to rule out cases where a

specification which is suppressed (i.e., it does not appear in any specification theory supported by the corresponding mutable specification theory) on account of a retracted specification reappears as a result of the retracted specification being discarded by a subsequent specification revision operation.

Example: Consider a mutable specification theory given by $W = \{\}$ and $D = \{\frac{:\neg a \wedge a \wedge (a \rightarrow b)}{\neg a}, \frac{:\top \wedge a \wedge (a \rightarrow b)}{\top}\}$. There is a single, empty, supported specification theory, corresponding to $Cn(\top)$, which is the only extension of (W, D). Let the specifcation base corresponding to this mutable specification theory consist of a single retracted specification $-(a \wedge (a \rightarrow b))$. Let us now retract b via an imperative retraction operation. The new mutable specification theory will be given by $W' = \{\}$ and $D' = \{\frac{:\neg a \wedge \neg b}{\neg a}, \frac{:\top \wedge \neg b}{\top}\}$. This default theory has one extension, $Cn(\neg a)$. Thus we get a mutable specification theory supporting a single specification theory consisting of the specification $\neg a$ as a result of retracting b from a mutable specification theory supporting a single, empty specification theory. This clearly violates the AGM contraction postulate which requires that the contracted belief set should be a subset of the original belief set, yet the behaviour is perfectly rational. The tentative specification $\neg a$ reappears in a specification theory as a consequence of the removal of the retracted specification that caused this tentative specification to be supressed.□

Clearly, only operations which do not display such behaviour can be related to the AGM framework.

Definition 4 *A specification revision operation $R((W, D), c_b, c_t, op, in)$ is called* conservative *iff $r \in c_b(Maxcomp(s_{inter}(op, s_{initial}((W, D)), in)))$ for any $r \in s_r(s_{initial}((W, D))$ for which there exists some $\frac{:\beta \wedge \gamma}{\beta} \in D$ where β and r are incompatible.*

We shall interpret postulate (1-), as one of way of articulating the following *principle of categorical matching* stated by Gärdenfors and Rott in [6] which requires that the representation of a belief state after a belief change has taken place should be of the same format as the representation of the belief state before the change. For postulates (2-) through (8-), we reformulate every condition on knowledge sets to apply to every extension of the PJ-default theory representing a mutable specification theory. For postulates (7-) and (8-) we can actually prove a stronger condition in the case that the antecedent in (8-) is satisfied. If the antecedent is not satisfied, there appears to be no obvious way to reformulate postulate (7-).

Theorem 2 *Let the following specification revision operations be imperative and conservative.*

- $R(\Delta_1, c_{b1}, c_{t1}, retraction, A) = (\Delta_2, t_2)$
- $R(\Delta_1, c_{b1}, c_{t1}, retraction, B) = (\Delta_3, t_3)$
- $R(\Delta_1, c_{b1}, c_{t1}, retraction, A \wedge B) = (\Delta_4, t_4)$
- $R(\Delta_2, c_{b2}, c_{t2}, addition, A) = (\Delta_5, t_5)$

These operations satisfy the following properties.

1. *The principle of categorical matching.*
2. *$\forall e' \in E(\Delta_2)$, there exists some $e \in E(\Delta_1))$ s.t. $e' \subseteq e$.*
3. *If $\not\models A$, then $\forall e : (e \in E(\Delta_2) \supset (e \not\models A)$.*
4. *If $\forall e' : (e' \in E(\Delta_1)) \supset (e' \models A)$ then for every $e' \in E(\Delta_1)$, there exists some e where $e \in E((\Delta_5))$ s.t $e' \subseteq e$.*
5. *If $\models A \leftrightarrow B$ then $E(\Delta_2) = E(\Delta_3)$.*

Our final result establishes that specifications are always recoverable in our framework.

Theorem 3 *If:*

- $\exists e : e \in E(\Delta)$ *s.t.* $e \models x$.
- $/\ \exists e : e \in E(\Delta')$ *s.t.* $e \models x$ *where* $R(\Delta, c_b, c_t, op_1, in_1) = (\Delta', t')$.
- $in_1 \notin c'_b(Maxcomp(s_{inter}(op_2, s_{initial}(\Delta), in_2)))$ *if* $op_1 = addition$.
- $-\neg in_1 \notin c'_b(Maxcomp(s_{inter}(op_2, s_{initial}(\Delta), in_2)))$ *if* $op_1 = retraction$.

then $\exists e : e \in E(\Delta'')$ *s.t.* $e \models x$ *where* $R(\Delta', c'_b, c'_t, op_2, in_2) = (\Delta'', t'')$.

Thus, if a specification x is contained in some specification theory supported by the initial mutable specification theory, but is therafter suppressed (i.e., it is not contained in any extension of the resulting mutable specification theory) on account of a revision operation op_1 involving input in_1, x will reappear if a later operation op_2 causes in_1 to be removed.

5 Conclusions

In this section, we shall summarize the major contributions of this work, and shall point out connections to related work. Our major contributions are as follows:

- We have presented a *language-independent* framework for revising formal specifications. To our knowledge, this is the first time such an effort has been attempted.
- We have presented a high-level method of analyzing formal specifications as *mutable specification theories*. Once again, this method is language-independent - any formal specification scheme can be analyzed in this fashion. By viewing mutable specification theories as default threories, we are able to apply a large corpus of results from AI to problems in software maintenance.
- We have argued for the use of a semantically well-founded notion of minimal change in specification revision to support reuse-oriented software maintenance. We have shown that the AGM framework for belief change provides a good starting point for such work.

- We have shown how mutable specification theories can be used to provide a compact and formally appealing representation of retracted specifications so that previously retracted specification are taken into account in determining candidate outcomes of the specification revision process. This represents a major departure from research in belief change inspired by the AGM framework.
- We have shown how specifications need not be discarded as a consequence of specification revision, but can be retained in a semantically well-founded manner in mutable specification theories. This is of special importance to reuse-oriented software maintenance.
- Our framework utilizes choice functions, like most other work in belief change. However, in a significant departure from existing methods, we treat the choice functions as inputs to the revision function, instead of coding the choice functions into the revision function itself. This permits context-specific choice, which is of crucial importance to context-driven software maintenance.
- Our appraoch to specification revision is non-prioritized; we do not accord any special priority to input specifications over existing ones. This makes our approach more practical than existing work in belief change, where the new belief is always accorded a higher priority over existing beliefs.
- An interesting consequence of our use of default logic is that the version we use (PJ-default logic) emerges as a formal specification language in its own right.
- Our approach is implementable. A prototype implementation is currently underway using the THEORIST system [15] (which effectively implements the subset of PJ-default logic we use in this work).

There has been previous work on using nonmonotonic representation formalisms for detecting inconsistencies in specifications during software maintenance [13] [2], but none of these have considered the problem of revising mutable specification theories using semantic notions from the AGM framework.

References

[1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] D. Cooke C.V. Ramamoorthy and C. Baral. Maintaining the truth of specifications in evolutionary software. *Int'l J. of Artificial Intelligence Tools*, 2:27–47, 1993.

[3] J. P. Delgrande and W. K. Jackson. Default logic revisited. In *Proc. of the Second International Conference on the Principles of Knowledge Representation and Reasoning*, pages 118–127, 1991.

[4] Peter Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge, MA, 1988.

[5] Peter Gärdenfors and David Makinson. Revisions of knowledge systems using epistemic entrenchment. In *Proc. of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 83–95, 1988.

[6] Peter Gärdenfors and Hans Rott. Belief revision. Technical Report 30, Zentrum Philosophie und Wissenschaftstheorie, Universität Konstanz, 1992.

[7] Aditya K. Ghose. *Practical belief change.* PhD thesis, Department of Computing Science, University of Alberta, 1995.

[8] Aditya K. Ghose and Randy Goebel. Default reasoning as belief change: A rationale. In *Proc. of the 2nd Pacific Rim Int'l Conf. on AI*, 1992.

[9] Aditya K. Ghose, Pablo O. Hadjinian, Abdul Sattar, Jia-H. You, and Randy Goebel. Iterated belief change: A preliminary report. In *Proc. of the Sixth Australian Joint Conference on AI*, 1993.

[10] Sven Ove Hansson. *Belief Base Dynamics.* Uppsala, 1991.

[11] W. L. Harper. Rational belief change, popper functions and counterfactuals. *Synthese*, (30):221–262, 1975.

[12] M. Lehman. Keynote address. In *IEEE CASE '90 4th Int'l Workshop on Computer-Aided Software Engineering.*

[13] Luqi and Daniel E. Cooke. How to combine nonmonotonic logic and rapid prototyping to help maintain software. *International Journal of Software Engineering and Knowledge Engineering*, 5:89–118, 1995.

[14] Abhaya C. Nayak. *Studies in Belief Change.* PhD thesis, Department of Philosophy, University of Rochester, 1993.

[15] David Poole, Randy Goebel, and Romas Aleliunas. Theorist: a logical reasoning system for defaults and diagnosis. In N.J. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–352. Springer Verlag, 1987.

[16] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1 and 2):81–132, 1980.